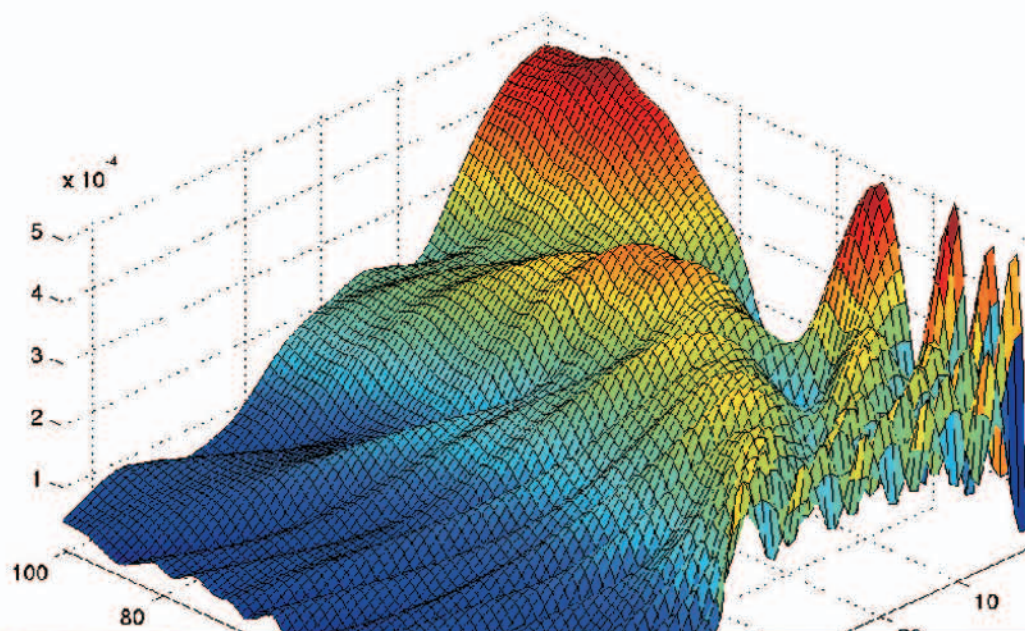
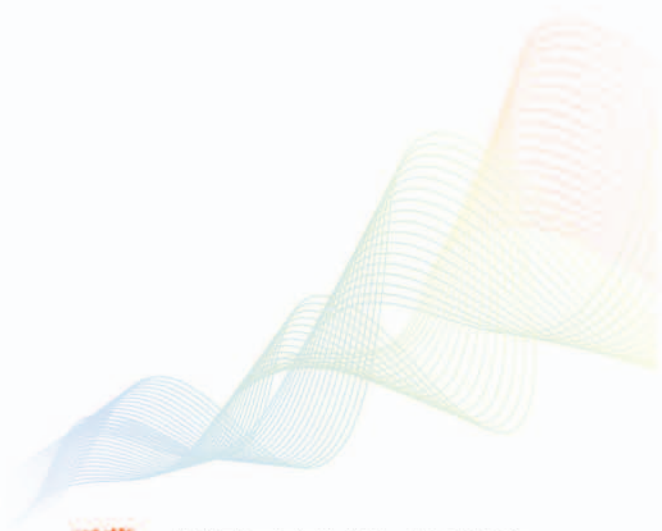


- 快速掌握MATLAB软件操作的基础知识
- 熟练应用MATLAB求解不同类型的问题
- 全面、系统地介绍MATLAB软件的各项功能



MATLAB

从基础到精通



王薇 毕业于南京农业大学，获硕士学位。

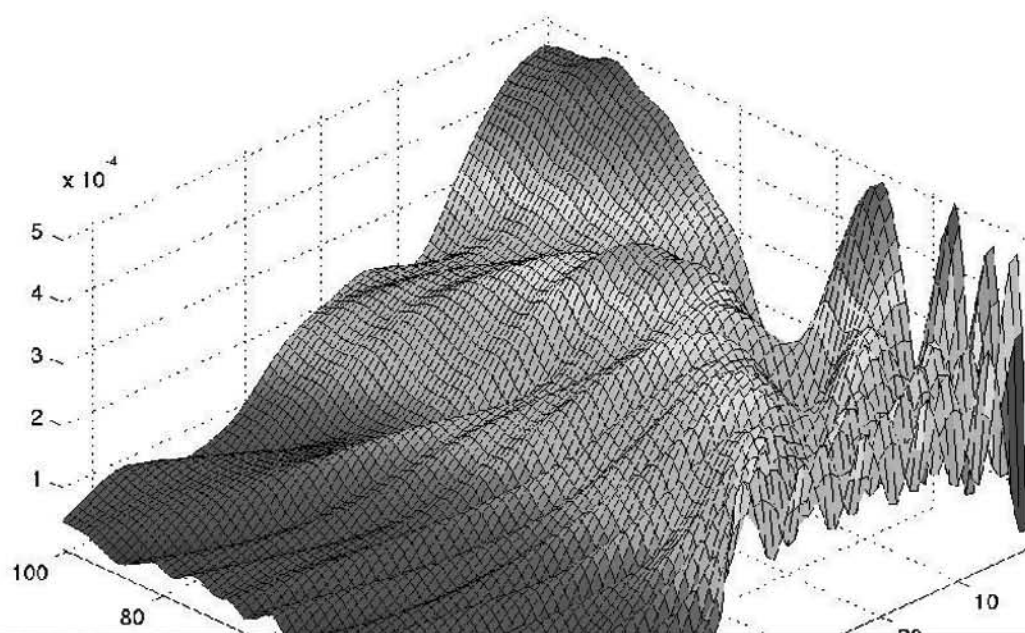
主要从事植被高光谱研究，使用MATLAB软件对海量的高光谱数据进行分析研究。参与过多项国家和省部级的科研项目。目前申请专利3项，获得软件著作权登记1项，发表论文多篇，其中SCI第一作者2篇，部分研究结果还被收录到最新的植被高光谱遥感的外文书中。



14小时多媒体教学视频

Broadview[®]
www.broadview.com.cn

- 快速掌握MATLAB软件操作的基础知识
- 熟练应用MATLAB求解不同类型的问题
- 全面、系统地介绍MATLAB软件的各项功能



MATLAB

从基础到精通



电子工业出版社
PUBLISHING HOUSE OF ELECTRONICS INDUSTRY
<http://www.phei.com.cn>

王薇◎编著

内 容 简 介

本书由浅入深地全面讲解了 MATLAB 软件的知识,以 MATLAB 7.0 版本的功能叙述为主。本书涉及面广,涵盖了一般用户需要使用的各种功能,并详细介绍了 MATLAB 常用工具箱的使用。在详细介绍 MATLAB 理论知识的同时,全程配合实例,使读者更容易掌握。本书附带 1 张 DVD 光盘,内容为本书多媒体语音教学视频及本书所涉及的源代码。

全书分为两篇。第 1 篇主要介绍 MATLAB 基础知识,第 2 篇主要介绍 MATLAB 常用工具箱的使用。涵盖的主要内容有 MATLAB 的发展、优势、特点和系统组成,常用的数据类型,矩阵和数组的基本操作,程序设计的相关知识,图形处理的相关知识,GUI 设计,数值分析技术,符号运算,常用的接口编程技术,文件的输入/输出机制,Simulink 的基础知识,统计工具箱,图像处理工具箱,优化工具箱,曲线拟合工具箱,神经网络工具箱,金融工具箱,小波分析工具箱,遗传算法与直接搜索工具箱等。

本书内容丰富,实例典型,实用性强,适合入门读者在较短的时间内有效地掌握 MATLAB 语言;对于广大 MATLAB 用户来说,也可以把本书当成一本常用的工具书使用。同时本书介绍了常用的专业工具箱的使用,因而也适用于相关专业的研究人士参考学习。

未经许可,不得以任何方式复制或抄袭本书之部分或全部内容。

版权所有,侵权必究。

图书在版编目(CIP)数据

MATLAB 从基础到精通 / 王薇编著. —北京:电子工业出版社, 2012.3

ISBN 978-7-121-15651-9

I. ①M… II. ①王… III. ①MATLAB 软件 IV. ①TP317

中国版本图书馆 CIP 数据核字(2011)第 282119 号

策划编辑:胡辛征

责任编辑:李云静

特约编辑:赵树刚

印 刷: 北京中新伟业印刷有限公司

装 订:

出版发行:电子工业出版社

北京市海淀区万寿路 173 信箱 邮编 100036

开 本: 787×1092 1/16 印张: 29.5 字数: 756 千字

印 次: 2012 年 3 月第 1 次印刷

印 数: 4000 册 定价: 65.00 元(含 DVD 光盘 1 张)

凡所购买电子工业出版社图书有缺损问题,请向购买书店调换。若书店售缺,请与本社发行部联系,联系及邮购电话:(010) 88254888。

质量投诉请发邮件至 zltz@phei.com.cn, 盗版侵权举报请发邮件至 dbqq@phei.com.cn。

服务热线:(010) 88258888。



前言

近年来, MATLAB 软件强大的数据计算和图形处理能力使其在各个领域得到了广泛的应用,越来越多的用户迫切需要尽快掌握 MATLAB 解决基本的问题。为此,目前市场上出现了大量讲述 MATLAB 使用的相关书籍。但是目前介绍 MATLAB 的书,其中一类主要是介绍某一功能、工具箱的使用,这类书一般只能对某个方向的相关内容进行详细阐述,在实际使用中读者仍需要参考 MATLAB 入门的相关书籍;而另一类主要是 MATLAB 基础性的相关书籍,对于 MATLAB 各种函数、工具箱,只做一些概念性的介绍,在实际使用中仍需要查阅庞大的 MATLAB 英文帮助文档。为此我们出版此书主要着眼于:

- ❑ 内容全面。完整介绍 MATLAB 各项功能,适合于各个层次的科学工作者,可以帮助入门读者快速掌握 MATLAB 基本操作,可以作为 MATLAB 使用者进一步提高操作能力的高效工具书。
- ❑ 容易掌握。对每个函数进行详细介绍,同时,结合实例分析讲解实际操作中可能遇到的问题。
- ❑ 面向需求。对常用工具箱进行详细的介绍,基本能满足解决各研究领域实际问题的需要。

本书有何特色?

1. 全面涵盖 MATLAB 的基础知识

本书为了便于读者能最大程度地掌握 MATLAB,包含了 MATLAB 的各项基础知识,从最基础的 MATLAB 软件的安装到常用的数值分析、图形处理、程序设计等内容,全部详细地介绍给读者,便于读者梳理、学习基础知识。

2. 详细讲述 MATLAB 的常用工具箱

MATLAB 工具箱的使用,可以为广大用户带来很多便利。MATLAB 工具箱功能强大,可用于解决神经网络、遗传算法、小波分析等复杂问题。本书向读者介绍了常用工具箱的使用,旨在使用户在较短的时间内掌握复杂算法的使用,而无须编写大量代码即可解决实际问题。

3. 配备实例,操作性强

本书在每一知识点讲述完成后都配备相应的实例,供读者演练,以使读者能较好地操作相应的知识点,同时这些源代码都收录在本书配套的光盘中,方便读者使用。

4. 附带教学视频,便于更好学习

本书附带相关章节的教学视频,帮助读者更好地学习本书的内容。

本书内容及知识体系

第 1 篇 基础知识（第 1~10 章）

本篇主要介绍了 MATLAB 的基础知识。

第 1 章主要总结 MATLAB 的发展、优势、特点和系统组成，以 MATLAB 7.0 为例介绍了软件的具体安装过程，并向广大读者介绍学习使用 MATLAB 的心得体会。

第 2 章介绍了 MATLAB 常用的数据类型，包括整型、浮点型、逻辑类型、结构体、元胞数组和字符串等的相关知识，以及数据之间的互相转换。

第 3 章主要讲述矩阵、数组的基本操作，涉及矩阵和数据的创建、简单运算、特殊运算、向量和高维数组的基本知识。

第 4 章讨论了 MATLAB 程序设计的相关知识，主要涉及程序设计的文件类型、变量和常量、流程控制、调试与优化等。

第 5 章主要介绍 MATLAB 图形处理的相关知识，涉及基本的绘图处理，二维、三维图形的绘制及图形编辑处理技巧。

第 6 章介绍了 MATLAB 创建 GUI 的两种方式，利用 GUI 向导和通过编程的方式。

第 7 章介绍了数值分析技术，包括简单的数据操作、多项式运算、微分和积分、拟合和插值、线性及非线性方程组的求解。

第 8 章重点讲述符号运算的内容，包括符号对象的创建、符号对象的常用操作和符号对象的转换。

第 9 章主要介绍常用的接口编程技术，包括 MATLAB 编辑器的使用、MEX 文件的使用、MAT 文件的使用、COM 组件技术与 Word、Excel 的混合使用技术。

第 10 章详细地介绍了 MATLAB 文件的输入/输出机制，包括 MATLAB 数据文件和图片文件的导入/导出等相关内容。

第 2 篇 常用工具箱使用（第 11~20 章）

本篇介绍了 MATLAB 常用工具箱的实现。

第 11 章主要介绍了 Simulink 的基础知识，包括 Simulink 的特点、Simulink 的建模环境、Simulink 的模型库和 Simulink 建模仿真的实现。

第 12 章主要探讨 MATLAB 统计工具箱在假设测验、方差分析、线性回归、非线性回归和多元统计等较为常用的统计问题中的应用。

第 13 章具体讲述图像处理工具箱的使用。对 MATLAB 中支持的图像文件格式、图像类型及其转换、图像处理工具箱如何完成基本的图像处理任务等做了具体的介绍。

第 14 章主要介绍 MATLAB 优化工具箱的使用，包括线性规划、整数规划、无约束规划和约束规划等常规的优化算法。

第 15 章主要介绍了 MATLAB 曲线拟合工具箱的使用，主要包括曲线拟合工具箱简介、利用 GUI 界面进行曲线拟合和利用命令行函数法进行曲线拟合。

第 16 章讨论了人工神经网络，包括 BP 神经网络、径向基神经网络、自组织神经网络、广义回归神经网络等网络算法在 MATLAB 神经网络工具箱中的使用。

第 17 章主要介绍了 MATLAB 金融工具箱的使用，主要内容为 MATLAB 金融工具箱的

组成和如何利用金融工具箱提供的函数进行常规的金融计算。

第 18 章涉及小波变换的基础知识，包括常用的小波分析操作和利用 GUI 实现小波分析等小波工具箱使用时需要具备的基础知识。

第 19 章主要介绍 MATLAB 遗传算法与直接搜索工具箱的使用。

第 20 章通过几个实例简单介绍 MATLAB 软件在数学建模、物理、化学等领域的应用。

配书光盘内容介绍

为了方便读者阅读本书，本书附带 1 张 DVD 光盘。内容如下：

- ☐ 本书主要实例的源代码。
- ☐ 本书主要内容的多媒体语音教学视频。
- ☐ 各章节内容的 PPT。

适合阅读本书的读者

- ☐ 零基础的 MATLAB 用户。
- ☐ 需要全面学习 MATLAB 的人员。
- ☐ 需要使用 MATLAB 提供的算法完成相关的程序设计。
- ☐ 需要在短时间内掌握 MATLAB 某些功能的各领域人员。
- ☐ 需要一本全面涵盖 MATLAB 各项内容查询手册的人员。

阅读本书的建议

- ☐ 没有 MATLAB 基础的读者，建议从第 1 章顺次阅读并练习每一个实例。
- ☐ 有一定 MATLAB 基础，且具有一定编程经验的读者，可以根据实际情况有重点地选择相关内容阅读，并注重实际的操作演练。
- ☐ 对于没有编程基础，希望快速使用 MATLAB 完成一些项目，以阅读其中的图形界面操作为主。

目 录

第 1 篇 基础知识

第 1 章	MATLAB 概述、安装和学习方法	2
1.1	MATLAB 简介	2
1.1.1	MATLAB 的发展历程	2
1.1.2	MATLAB 的优势和特点	3
1.1.3	MATLAB 的系统组成	3
1.2	MATLAB 7.0 的安装	4
1.3	MATLAB 用户界面	6
1.3.1	启动和退出	6
1.3.2	主菜单	7
1.3.3	标题栏	17
1.3.4	命令窗口	17
1.3.5	当前目录浏览窗口	19
1.3.6	工作空间浏览窗口	22
1.3.7	历史命令窗口	24
1.4	帮助系统	24
1.4.1	帮助浏览器	25
1.4.2	命令帮助系统	27
1.4.3	远程帮助系统	28
1.5	如何学习 MATLAB	28
1.6	本章小结	29
第 2 章	MATLAB 的数据类型	30
2.1	整型	30
2.2	浮点型	31
2.3	逻辑类型	32
2.4	字符串	33
2.4.1	字符串的生成	33
2.4.2	字符串操作函数	34
2.5	元胞数组	37
2.5.1	元胞数组的创建	37
2.5.2	元胞数组的访问	38
2.5.3	元胞数组的显示	39
2.5.4	元胞数组的删除	40

2.6	结构体	41
2.6.1	结构体的生成	41
2.6.2	结构体的操作	42
2.7	不同数据类型之间的转化	43
2.8	本章小结	46
第 3 章	矩阵和数组	47
3.1	矩阵和数组的概念	47
3.2	矩阵和数组的创建	47
3.3	矩阵及数组的基本操作	51
3.3.1	基本信息获取	51
3.3.2	元素访问	53
3.4	矩阵及数组的简单运算	57
3.4.1	基本函数	57
3.4.2	加减运算	63
3.4.3	乘法运算	64
3.4.4	除法运算	64
3.4.5	乘方运算	66
3.5	矩阵的特殊运算	67
3.5.1	行列式运算	67
3.5.2	逆运算	67
3.5.3	秩运算	68
3.5.4	特征值运算	68
3.6	数组的特殊运算	68
3.6.1	关系运算	68
3.6.2	逻辑运算	71
3.7	向量及其运算	73
3.7.1	向量的生成	73
3.7.2	向量的运算	73
3.8	高维数组操作	78
3.8.1	高维数组的创建	78
3.8.2	高维数组的基本操作	80
3.9	本章小结	83
第 4 章	程序设计	84
4.1	程序设计概述	84
4.2	脚本文件	87
4.3	函数文件	87
4.3.1	函数的定义	87

4.3.2	函数类型	89
4.3.3	函数的调用和变量传递	90
4.3.4	输入/输出参数的控制	91
4.4	常量、变量	92
4.4.1	变量的命名	92
4.4.2	系统预定义的常量	92
4.4.3	变量类型	93
4.4.4	系统预定义的变量	93
4.5	程序结构及流程控制	94
4.5.1	赋值语句	94
4.5.2	条件语句	94
4.5.3	循环语句	98
4.6	交互控制指令	101
4.7	程序的调试	106
4.7.1	常见错误类型	106
4.7.2	调试方法	108
4.8	优化	110
4.8.1	循环的向量化	110
4.8.2	循环的优化	111
4.8.3	M 文件分析	111
4.8.4	提高编程效率的小技巧	113
4.9	本章小结	114
第 5 章	图形处理	115
5.1	基本的绘图处理	115
5.1.1	常用函数	115
5.1.2	MATLAB 图形窗口	124
5.1.3	坐标控制	127
5.1.4	图形标注	130
5.1.5	窗口分割	131
5.1.6	MATLAB 图形编辑工具的使用	132
5.2	特殊二维图形	140
5.2.1	条形图	140
5.2.2	直方图	142
5.2.3	面积图	143
5.2.4	饼图	145
5.2.5	散点图	145
5.2.6	排列图	146

5.2.7 罗盘图	147
5.2.8 羽毛图	148
5.2.9 矢量图	148
5.2.10 杆型图	149
5.2.11 阶梯图	150
5.2.12 极坐标图	151
5.2.13 等值线图	152
5.2.14 曲线误差的添加	153
5.3 三维图形	154
5.3.1 三维图形的绘制	154
5.3.2 三维图形的编辑	162
5.4 本章小结	168
第 6 章 图形用户界面 (GUI)	169
6.1 GUI 简介	169
6.1.1 GUI 的创建方法概述	169
6.1.2 GUI 的设计流程	170
6.1.3 GUI 界面设计的原则	170
6.2 利用 GUIDE 设计 GUI	170
6.2.1 新建 GUI 设计界面	171
6.2.2 常用控件的设计	173
6.2.3 界面设计窗口的常用工具	174
6.3 利用程序语言设计 GUI	178
6.3.1 GUI 对象编程	178
6.3.2 GUI 的对话框	181
6.4 GUI 文件	190
6.4.1 M 文件结构	190
6.4.2 回调函数	191
6.4.3 参数的传递	192
6.5 GUI 界面设计的实例	193
6.6 本章小结	198
第 7 章 数值分析	199
7.1 简单的数据操作	199
7.1.1 随机数的生成	199
7.1.2 描述性统计参数的计算	206
7.1.3 描述性统计做图	212
7.1.4 数据的排序	215
7.2 多项式运算	216



7.2.1	多项式求值	216
7.2.2	多项式求根	217
7.2.3	多项式乘除	217
7.2.4	多项式微积分	218
7.3	微分和积分	218
7.3.1	数值微分	218
7.3.2	数值积分	219
7.4	拟合和插值	220
7.4.1	拟合基础	220
7.4.2	线性拟合	220
7.4.3	非线性拟合	221
7.4.4	插值基础	222
7.5	线性方程组的求解	227
7.6	非线性方程求解	227
7.7	本章小结	229
第 8 章	符号计算功能	230
8.1	符号计算概述	230
8.2	符号对象的创建	230
8.2.1	符号变量	230
8.2.2	符号常量	231
8.2.3	符号表达式	232
8.2.4	符号矩阵	232
8.2.5	符号函数	233
8.3	符号表达式的基本操作	233
8.3.1	符号表达式的化简	233
8.3.2	符号表达式的合并与分解	235
8.3.3	符号表达式的代数运算	236
8.3.4	符号表达式的分子分母提取	236
8.3.5	符号表达式的自变量的确定	236
8.4	符号矩阵运算	237
8.4.1	符号矩阵的代数运算	237
8.4.2	符号矩阵的特殊运算	238
8.5	符号微积分运算	239
8.5.1	符号极限	239
8.5.2	符号微分	239
8.5.3	符号积分	240
8.5.4	符号级数	240

8.6	符号方程求解	241
8.6.1	代数方程的求解	241
8.6.2	微分方程的求解	241
8.7	符号函数图形绘制	242
8.7.1	函数 ezplot()	242
8.7.2	函数 fplot()	243
8.7.3	函数 ezplot3()	243
8.7.4	函数 ezcontour()	244
8.7.5	函数 ezcontourf()	244
8.7.6	函数 ezmesh()	245
8.7.7	函数 ezmeshc()	245
8.7.8	函数 ezpolar()	246
8.8	符号对象与数值对象的转换	246
8.8.1	符号对象转换为数值对象	246
8.8.2	数值对象转换为符号对象	247
8.9	本章小结	247
第 9 章	应用程序接口	248
9.1	MATLAB 编译器	248
9.1.1	编译器简介	248
9.1.2	编译器的安装、配置	249
9.1.3	编译器的使用	250
9.2	MEX 文件	250
9.3	MAT 文件	252
9.4	MATLAB 引擎技术	254
9.5	COM 组件	256
9.6	与 Word、Excel 的混合使用	259
9.6.1	Excel Link 的使用	259
9.6.2	在 Word 中使用 Notebook	263
9.7	本章小结	266
第 10 章	文件 I/O	267
10.1	数据文件	267
10.1.1	低级文件的 I/O 操作	267
10.1.2	高级文件的 I/O 操作	272
10.1.3	利用界面工具导入/导出数据	279
10.2	图片文件	281
10.2.1	不同格式图片文件的导入	281
10.2.2	不同格式图片文件的导出	281
10.3	本章小结	282

第 2 篇 常用工具箱使用

第 11 章 Simulink 仿真	284
11.1 Simulink 简介	284
11.1.1 Simulink 特点	284
11.1.2 Simulink 工作环境	285
11.2 Simulink 常用基本模块	290
11.3 Simulink 建模与仿真	302
11.3.1 选择模块	302
11.3.2 模块基本操作	303
11.3.3 信号线操作	304
11.3.4 参数设置	305
11.3.5 仿真运行	310
11.3.6 模型仿真举例	310
11.4 本章小结	311
第 12 章 统计工具箱	312
12.1 假设测验	312
12.1.1 单个正态总体的假设测验	312
12.1.2 两个正态总体的假设测验	313
12.2 方差分析	314
12.2.1 单因素方差分析	314
12.2.2 双因素方差分析	318
12.2.3 多因素方差分析	319
12.3 线性回归	321
12.4 非线性回归	325
12.5 多元统计	327
12.5.1 判别分析	327
12.5.2 聚类分析	328
12.5.3 主成分分析	331
12.5.4 因子分析	333
12.6 本章小结	334
第 13 章 图像处理工具箱	335
13.1 图像文件格式	335
13.2 图像类型及其转换	336
13.2.1 真彩色图像 (RGB images)	336
13.2.2 索引图像 (Index images)	337
13.2.3 灰度图像 (Intensity images)	338
13.2.4 二值图像 (Binary images)	338

13.2.5 图像类型转换.....	339
13.3 图像处理的基本操作.....	344
13.3.1 图像读入和显示.....	344
13.3.2 图像缩放、旋转、剪切.....	350
13.3.3 图像的代数运算.....	353
13.3.4 图像增强.....	356
13.3.5 图像变换.....	358
13.3.6 图像滤波.....	360
13.4 本章小结.....	362
第 14 章 优化工具箱.....	363
14.1 线性规划.....	363
14.2 整数规划.....	365
14.3 二次规划.....	366
14.4 非线性规划.....	367
14.4.1 无约束非线性规划.....	367
14.4.2 约束非线性规划.....	369
14.5 本章小结.....	372
第 15 章 曲线拟合工具箱.....	373
15.1 曲线拟合工具箱简介.....	373
15.2 利用图形界面进行曲线拟合.....	373
15.2.1 打开曲线拟合工具箱.....	373
15.2.2 导入拟合数据集.....	374
15.2.3 拟合数据预处理.....	375
15.2.4 曲线拟合.....	376
15.2.5 异常数据的去除.....	380
15.2.6 绘图显示设置.....	380
15.3 使用命令行拟合数据.....	381
15.4 本章小结.....	383
第 16 章 神经网络工具箱.....	384
16.1 人工神经网络介绍.....	384
16.1.1 人工神经网络的基本特征.....	384
16.1.2 人工神经网络的分类.....	385
16.1.3 人工神经网络的应用.....	385
16.1.4 MATLAB 人工神经网络工具箱.....	386
16.2 BP 神经网络.....	386
16.2.1 BP 神经网络基本原理.....	386
16.2.2 BP 神经网络的特点.....	386

16.2.3	BP 神经网络的应用	387
16.2.4	BP 神经网络的实现	387
16.2.5	BP 神经网络的不足	390
16.3	径向基神经网络	391
16.3.1	径向基神经网络的基本原理	391
16.3.2	径向基神经网络的实现	391
16.4	广义回归神经网络	392
16.5	自组织神经网络	393
16.6	神经网络的 GUI 界面实现	393
16.7	本章小结	399
第 17 章	金融工具箱	400
17.1	金融工具箱简介	400
17.2	金融数据的获取和可视化	401
17.2.1	金融数据的获取	401
17.2.2	金融数据的可视化	406
17.3	金融数据分析	408
17.3.1	投资组合分析	408
17.3.2	利率期限计算	408
17.3.3	资金流量估算	410
17.3.4	时间序列分析	411
17.4	本章小结	412
第 18 章	小波分析工具箱	413
18.1	小波变换的基础知识	413
18.2	常用的小波分析操作	415
18.2.1	一维小波分析	415
18.2.2	二维小波分析	419
18.2.3	小波包变换	422
18.2.4	信号去噪	425
18.2.5	信号压缩	428
18.3	利用 GUI 实现小波分析	428
18.3.1	小波分析工具箱 GUI 的启动	429
18.3.2	小波分析工具箱 GUI 的工作界面	429
18.3.3	小波分析工具箱的操作	430
18.4	本章小结	436
第 19 章	遗传算法工具箱	437
19.1	遗传算法的基本概念	437
19.2	遗传算法工具箱	438

19.2.1 遗传算法相关函数.....	438
19.2.2 利用 GUI 实现遗传算法.....	443
19.3 直接搜索工具箱	445
19.3.1 利用命令行方式实现直接搜索	446
19.3.2 利用 GUI 方式实现模式搜索	447
19.4 本章小结	448
第 20 章 MATLAB 在各领域的应用	449
20.1 MATLAB 在数学建模中的应用	449
20.2 MATLAB 在物理中的应用	451
20.3 MATLAB 在化学中的应用	453
20.4 MATLAB 在生命科学中的应用	454
20.5 MATLAB 在社会科学中的应用	455
20.6 本章小结	456

第 1 篇

基础知识

- 第 1 章 MATLAB 概述、安装和学习方法
- 第 2 章 MATLAB 的数据类型
- 第 3 章 矩阵和数组
- 第 4 章 程序设计
- 第 5 章 图形处理
- 第 6 章 图形用户界面 (GUI)
- 第 7 章 数值分析
- 第 8 章 符号计算功能
- 第 9 章 应用程序接口
- 第 10 章 文件 I/O



第1章

MATLAB 概述、安装和 学习方法

本章主要介绍在使用和学习 MATLAB 软件前需要掌握的一些基础知识。首先总结 MATLAB 的发展、优势、特点和系统组成,使读者对 MATLAB 软件总体有一定把握。在此基础上本书以 MATLAB 7.0 为例分别介绍 MATLAB 软件的具体安装过程。同时本章将对 MATLAB 7.0 各主要用户界面功能做详细的介绍,用户界面为使用者提供了友好的操作平台;而对 MATLAB 帮助系统的了解,将为读者在实际工作中遇到的问题提供有效的指导。最后笔者结合前人和自己使用 MATLAB 的多年经验,向广大读者介绍学习使用 MATLAB 的心得体会,希望读者能掌握学习使用 MATLAB 的有效方法。本章是学习使用 MATLAB 软件最为基础的章节,是后续章节有效学习的基石。

1.1 MATLAB 简介

MATLAB 是由美国 Mathworks 公司开发的用于数据分析处理、算法开发应用、数据可视化及图形处理的科学软件。使用 MATLAB 产品,相比传统编程语言,例如 C、C++和 Fortran 等,能更为快速地解决技术计算的问题,因而 MATLAB 被越来越广泛地应用于科学计算、嵌入式系统、控制系统、数字信号处理、图像和视频处理、通信系统、计算金融学等领域。

MATLAB 是新一代的高性能计算语言。它集成了计算、可视化和编程在一个友好的环境中,问题和解决方案都是以熟悉的数学符号表示。典型用途包括如下几种。

- 数学和计算。
- 算法开发。
- 建模和仿真。
- 数据分析、挖掘和可视化。
- 科学和工程制图。
- 图形用户界面的应用开发。

1.1.1 MATLAB 的发展历程

美国新墨西哥大学计算机科学系主任 Cleve Moler 和他的同事于 20 世纪 70 年代编写了 MATLAB,最早的 MATLAB 是用 Fortran 语言编写的,早期版本以简单矩阵运算为主,主要是为了减轻学生编程的负担。MATLAB 为矩阵 (Matrix) 和实验室 (LABoratory) 前几个英文字母组合。随后在 1984 年 Little、Moler 等人共同创立了 MathWorks 公司,正式把 MATLAB 产品推向市场。此后 MathWorks 公司不断完善产品,图形图像处理、符号运算等多种功能被添加到 MATLAB 中,同时随着信息技术、科学算法的不断发展,不同领域中需要使用的算法也被集成为工具箱加入 MATLAB。目前, MATLAB 已成为国际公认的优秀计算软件,它在各领域的应用也越来越广,并且有着更广阔的应用前景。

1.1.2 MATLAB 的优势和特点

近年来 MATLAB 在各领域的应用越来越广泛，这主要得益于 MATLAB 具有以下优势。

1. 编程效率高

对于复杂算法的开发，MATLAB 函数库提供了大量函数可供直接调用，面向专业领域的工具箱减轻了以往需要复杂编程开发的负担，相比一般编程语言可大大节省算法开发时间，而对于非算法研究领域人员，如果只想在自己的领域使用已有的算法，可灵活地调用 MATLAB 编写好的函数。

2. 使用方便

MATLAB 多数工具箱提供了图形界面操作，无须编写代码，直接通过界面操作。

3. 扩充能力好，移植性强

通过 MATLAB 编写的程序可通过多种途径与常用的编程语言、应用软件连接，扩充移植能力强。

4. 开放性好

MATLAB 一般函数都是以 m 文件形式存在的，可以直接打开相应算法的 m 文件，查阅源代码。

5. 简单易用的程序语言

MATLAB 语言特征与 C 语言相似，而且更加简单，更加符合专业人员对算法的书写格式。

6. 高效方便的矩阵运算

MATLAB 的基本数据单位是矩阵，与数学、工程中常用的形式十分相似，故用 MATLAB 来解决问题较为简单。

7. 方便的绘图功能

MATLAB 提供了一系列绘图函数，可方便地对数据进行可视化分析。

下面简单介绍 MATLAB 主要特点。

- 科学计算的高级语言，为算法开发提供高效的实现途径。
- 提供完善的开发环境，系统管理代码、文件、数据。
- 提供友好的交互界面，用于探索、设计、解决实际问题。
- 内含大量的可直接使用的数学函数，涉及线性代数、统计学、傅里叶分析、滤波、优化及数值积分等方面。
- 提供二维和三维图用于数据的可视化分析。
- 可以自定义生成图形用户界面。
- 编写的代码可与外部的程序和其他编程语言集成，例如 C、C++、Fortran、Java 和 Microsoft Excel 等。

1.1.3 MATLAB 的系统组成

MATLAB 主要由以下几部分组成。

1. 开发环境

开发环境帮助使用者高效地处理 MATLAB 函数和文件，其中许多环境提供了友好的图形界面，可交互使用，主要包括：命令窗口，代码编辑、调试、分析窗口，工作空间窗口，浏览器查看帮助窗口。

2. 数学函数库

数学函数库包含了从初等函数（如求和、正弦、余弦）到更为复杂的算法（如矩阵求逆、矩

阵特征值、快速傅里叶变换)在内的大量数学函数。

3. MATLAB 语言

MATLAB 语言是高级矩阵/数组语言,具有流程控制、函数调用、数据结构、输入/输出、面向对象编程等程序语言特征。利用 MATLAB 语言既可以进行小规模程序设计,快速完成算法设计的基本任务,也可进行大规模编程,开发复杂的程序算法。

4. 图形功能

MATLAB 提供了图形功能,可视化地显示向量和矩阵,同时提供在图形上注释和打印功能。此外还包括二维和三维数据可视化、图像处理、动画生成、演示图形等高级功能;用户自定义图形对象,建立完整的图形用户界面等低级功能。

5. 应用程序接口

利用 MATLAB 提供的外部应用程序可以开发 C/C++和 Fortran 与 MATLAB 交互的程序算法。包括将 MATLAB 作为计算引擎的 MATLAB 引擎技术,建立动态链接调用 MATLAB 程序文件和 MAT 数据文件读/写。

综上可以看出, MATLAB 是一个功能强大的系统,集数据计算、图形管理、程序开发为一体,向用户提供了友好的操作界面环境,同时通过与其他编程环境的交互操作,可以充分利用 MATLAB 语言的优势,大大缩短算法开发周期。

1.2 MATLAB 7.0 的安装

在购买 MATLAB 7.0 后,可以按照相关的说明进行安装,安装过程比较简单。安装前需要准备好购买正版软件所获得的授权注册码,此后可按以下步骤安装 MATLAB 7.0。

(1) 开始安装:将 MATLAB 7.0 安装盘放入光驱,打开 MATLAB 7.0 安装盘,双击按钮 , 开始安装过程。初始化后打开“Welcome to the MathWorks Installer”窗口,如图 1.1 所示。选择“Install”单选按钮,单击“Next”按钮进入下一步的安装。对于最初使用 MATLAB 试用版的用户,在购买正式版后,安装时选择“Update license without installing anything, using a new PLP”单选按钮,可以更新序列号,无须重新安装软件。

(2) 用户信息登记和授权注册码输入:在步骤(1)中单击“Next”按钮后,将会打开“License Information”窗口,如图 1.2 所示。在此窗口中需要输入用户信息和授权注册码。分别在“Name”和“Company”文本框中填入用户姓名和公司名称,并在相应位置输入授权注册码(PLP),授权注册码用于验证是否有权安装 MATLAB。

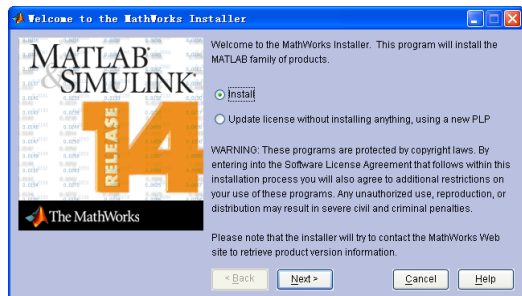


图 1.1 “Welcome to the MathWorks Installer”窗口

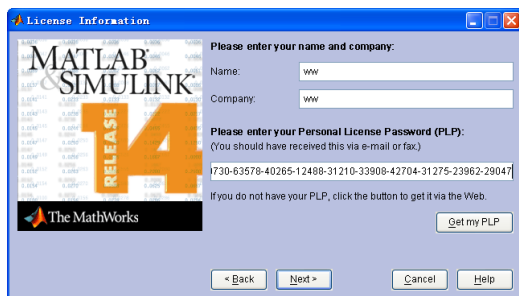


图 1.2 “License Information”窗口

(3) 软件用户协议: 如果上一步中输入授权注册码正确, 单击“Next”按钮打开“Licensing Agreement”窗口, 如图 1.3 所示。该窗口中的内容为软件用户使用的协定, 阅读后如果同意协定, 就选择“Yes”单选按钮并单击“Next”按钮进入下一步操作。

(4) 安装方式选择: MATLAB 提供了两种安装方式, 典型安装(Typical)和自定义安装(Custom), 如图 1.4 所示。选择 Typical 方式安装, 将会按软件默认的设置将所有用户购买的产品按照默认方式安装; 选择 Custom 方式安装, 可以自行设定需要安装的组件。对于入门的读者建议选择典型安装方式。继续单击“Next”按钮进入下一步操作。



图 1.3 “Licensing Agreement” 窗口

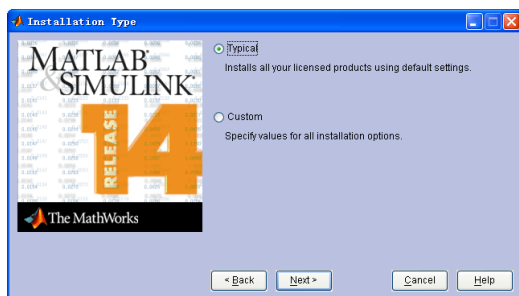


图 1.4 “Installation Type” 窗口

(5) 软件安装路径选择: 在打开的“Folder Selection”窗口中选择软件的安装目录, 单击“Browse”按钮可以浏览选择计算机上的安装目录。

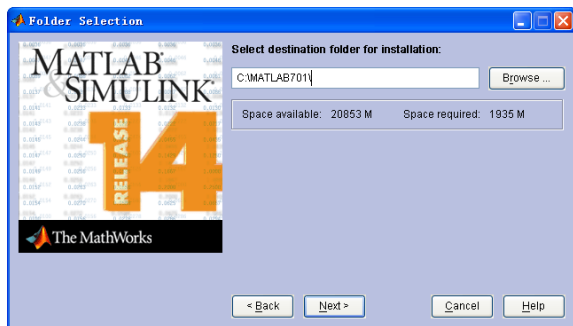


图 1.5 “Folder Selection” 窗口

(6) 确认安装: 在 MATLAB 开始复制文件到硬盘之前, 如图 1.6 所示, 会打开“Confirmation”窗口给出安装说明, 包括软件安装的目录、安装的产品等。如果软件安装设置有问题, 单击“Back”按钮返回之前的过程, 重新选择、设置安装过程。如果确认无误, 单击“Install”按钮确认安装, 之后将弹出如图 1.7 所示的安装进度框。

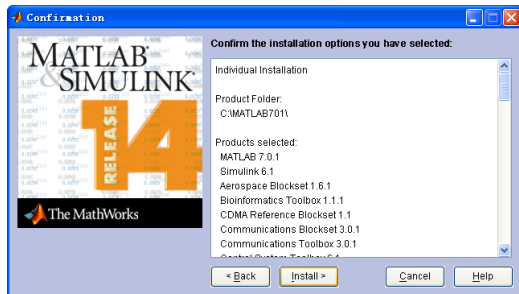


图 1.6 “Confirmation” 窗口

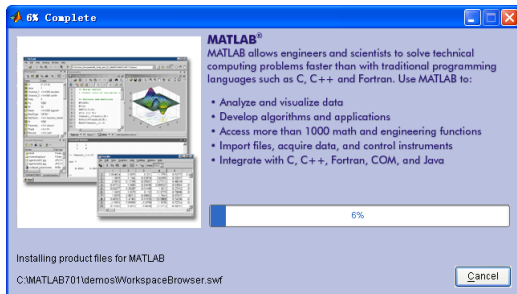


图 1.7 安装进度框

(7) 阅读产品配置的注意事项：安装完成后，会打开如图 1.8 所示的产品配置注意事项的窗口，主要根据目前已安装的 MATLAB 产品，提供产品配置信息和可用产品更新，告诉用户目前所安装的产品是否有需要额外配置。如有可用产品更新，在安装过程中，安装程序将尝试联系 MathWorks 公司网站，以确定是否有一个或更多更新的产品版本可供下载。

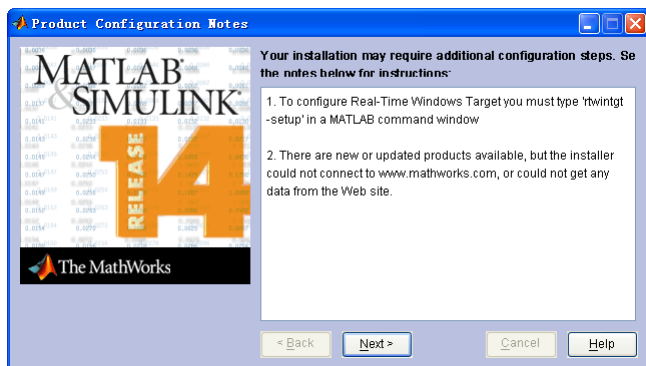


图 1.8 “Product Configuration Notes” 窗口

(8) 安装完成：MATLAB 安装程序完成时，会显示如图 1.9 所示安装完成窗口。在此窗口中，可以选择在退出安装程序时启动 MATLAB。如果不想立即启动 MATLAB，取消“Start MATLAB”复选框中选择。最后单击“Finish”按钮结束安装过程。

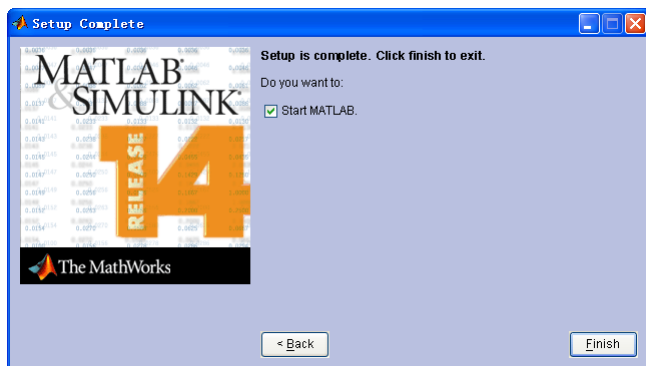


图 1.9 “Setup Complete” 窗口

1.3 MATLAB 用户界面

本节主要介绍 MATLAB 各用户界面中各窗格的功能和使用。用户界面是利用 MATLAB 进行各项操作的基本环境，了解和掌握各用户界面的功能将有助于更好地处理、解决问题。

1.3.1 启动和退出

MATLAB 的启动与一般常用软件的启动方式类似，主要有以下 3 种启动方式：

- (1) 双击桌面上的 MATLAB 按钮 ，通过快捷方式打开程序。
- (2) 选择“开始”→“所有程序”菜单中的 MATLAB 7.0 的可执行程序。
- (3) 在 MATLAB 的安装路径中找到 MATLAB 的可执行程序 MATLAB.exe。

在启动 MATLAB 程序后，将打开如图 1.10 所示的主界面，该界面主要由主菜单、工作空间

浏览窗格、当前目录浏览窗格、历史命令窗格等组成。

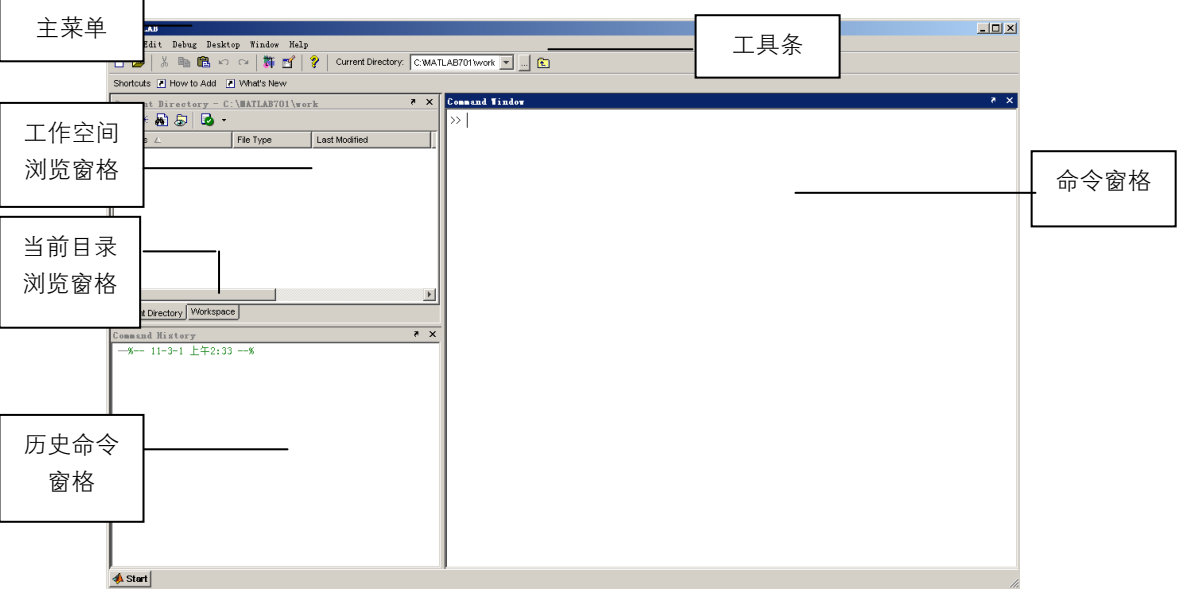


图 1.10 MATLAB 7.0 主界面

退出 MATLAB 7.0 最简单的方法就是直接单击主界面上的关闭按钮，除此之外也可以通过选择“File”→“Exit MATLAB”命令退出，或者按【Ctrl+Q】组合键，或者在命令窗格输入“quit”命令。

1.3.2 主菜单

MATLAB 7.0 的主菜单主要包括“File”、“Edit”、“Debug”、“Desktop”、“Help”等子菜单项，本节将详细讲述主要菜单项的功能和使用。

1. File 子菜单

File 子菜单主要包含新建/打开文件、关闭窗口、导入数据、保存工作空间内的数据、设置 MATLAB 的搜索路径、软件属性设置、打印及其页面设置、退出 MATLAB 等选项，如图 1.11 所示。

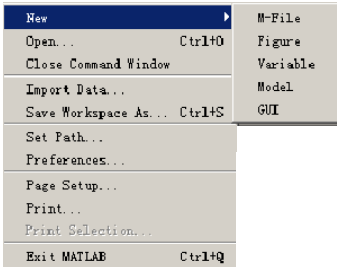


图 1.11 File 菜单

其中，

- “Import Data”菜单项：向 MATLAB 工作空间导入数据。

选择“File”→“Import Data”命令，再选择导入的数据文件，本例中选择 C:\MATLAB701\toolbox\stats 目录下的 gas 数据文件导入，弹出如图 1.12 所示的“Import Wizard”窗口。该数据集有 price1 和 price2 两个变量，单击“Finsh”按钮完成数据导入工作，在 MATLAB

工作空间中将出现变量 price1 和 price2。

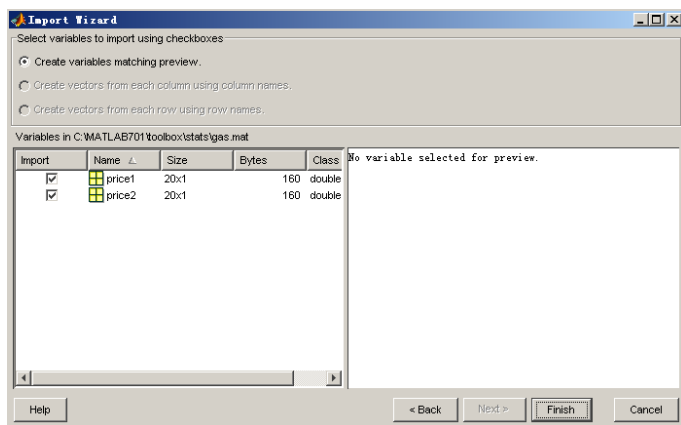


图 1.12 “Import Wizard” 窗口

- “Save Workspace As” 菜单项：用于以 MATLAB 的数据格式（mat 形式）保存工作空间内的数据。
- “Set Path” 菜单项：设置 MATLAB 的搜索路径。

搜索路径是 MATLAB 可以识别查找的所有文件的路径。默认情况下 MATLAB 安装目录中的所有工具箱子文件夹和 work 文件夹都在 MATLAB 的搜索路径下。设置搜索路径机制就是将一些可能要用到的函数或文件的存放路径设置在 MATLAB 的搜索路径中。当在搜索路径中含有文件名相同的文件时，搜索路径顺序是十分重要的，MATLAB 将优先使用搜索路径中最靠前的文件夹下的文件。因此设置搜索路径将有助于找到真正需要的文件。MATLAB 搜索路径信息保存在 pathdef.m 文件中，pathdef.m 文件内容为搜索路径中每个文件夹的全路径名，中间用“;”分隔，默认情况下 pathdef.m 文件是在 matlabroot/toolbox/local 路径下。

选择“File”→“Set Path”命令，打开如图 1.13 所示的“Set Path”窗口。该窗口主要提供了添加、删除文件搜索路径，改变搜索路径顺序，保存设置搜索路径等功能。其中添加、删除文件搜索路径主要是在用户安装、卸载工具箱时设置，将包含所需安装工具箱 m 文件的目录通过“Set Path”对话框中的“Add Folder”或“Add with Subfolder”按钮加入搜索路径即完成工具箱的安装，“Remove”按钮用于删除搜索路径。

当用户在命令窗格中输入命令后，MATLAB 会按照设置的搜索路径顺序查找该命令，执行第一个查找到的命令。如果在命令窗格输入“xpath”命令，MATLAB 会按以下顺序搜索命令：

（1）在 MATLAB 工作空间窗格中是否存在变量 xpath，如果存在显示该变量，不存在则转入下一步搜索。

（2）是否存在函数名为 xpath 的子函数，如果有则调用此函数，没有则转入下一步搜索。

（3）是否存在函数名为 xpath 的私有函数，如果有则调用此函数，没有则转入下一步搜索。

（4）是否存在函数名为 xpath 的重载函数，如果有则调用此函数，没有则转入下一步搜索。

（5）在当前目录中是否存在函数名为 xpath 的 m 文件，如果有则调用此函数，没有则转入下一步搜索。

（6）在搜索路径中是否存在函数名为 xpath 的 m 文件，按照搜索路径中的路径顺序依次查询。这里特别需要注意，在添加了工具箱路径后，调用工具箱中的函数有时会出现所用用法、调用格式都没有错，但是程序仍然出错的现象，这很可能是因为 MATLAB 其他工具箱中有同名的函数，而该工具箱的搜索路径顺序在所安装工具箱路径顺序之前。

如果在经过上述步骤的搜索后，仍然没有找到名为“xpath”的变量或函数，则 MATLAB 会报错：“??? Undefined function or variable 'xpath'”。

搜索路径的保存路径更改后，如果仅希望在本次使用 MATLAB 中生效，单击“Close”按钮，弹出如图 1.14 所示的“Save Path”对话框，单击“否 (N)”按钮。如果希望设置的搜索路径关闭 MATLAB 下次重启后仍然有效，单击“Save Path”对话框中的“是 (Yes)”按钮或者直接单击“Save”按钮保存路径。

“Save Path”窗口中的“Default”按钮用于恢复搜索路径为默认状态。

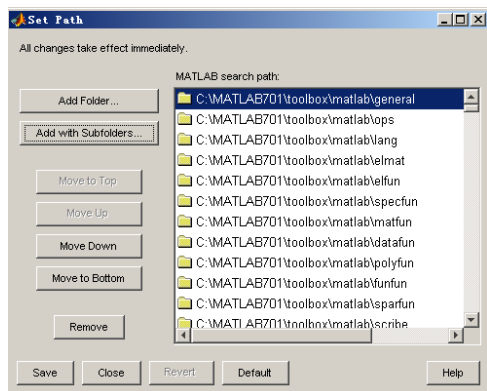


图 1.13 “Set Path” 窗口

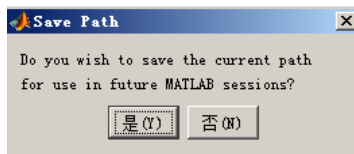


图 1.14 “Save Path” 对话框

- “Preferences” 菜单项：用于设置 MATLAB 系统的参数。

选择“File”→“Preferences”命令，打开如图 1.15 所示的“Preferences”窗口，从窗口左半边的树形结构图可以看出，主要为字体、各窗口的一些参数设置。其中常用的几项参数设置如下。

字体设置：单击树形目录中的“Fonts”选项，显示如图 1.16 所示的字体设置窗口，主要包括代码和文本字体类型、样式、大小设置，可以根据使用者需要设计为适合阅读、书写的形式。

命令窗格显示设置：单击树形目录中的“Command Window”选项，显示如图 1.17 所示的命令窗格输出设置窗口。其中 Text display 区域用于设置输出数据形式和输出间距控制，输出数据形式通过 Numeric format 下拉列表选择，设置输出数据格式，但不影响计算精度；输出数据间距通过 Numeric display 下拉列表选择，选择“compact”设置紧凑的格式显示数据，固定大小的窗口可多显示一些代码；选择“loose”设置松散的格式显示数据，便于阅读。以下为设置不同输出数据显示格式的例子。

- (1) 设置 Numeric format: short; Numeric display: loose。

```
>> 5/3  
  
ans =  
  
1.6667
```

- (2) 设置 Numeric format: long; Numeric display: compact。

```
>> 5/3  
ans =  
1.666666666666667
```

- (3) 设置 Numeric format: short e; Numeric display: compact。

```
>> 5/3
ans =
    1.6667e+000
```

(4) 设置 Numeric format: long e; Numeric display: compact。

```
>> 5/3
ans =
    1.666666666666667e+000
```

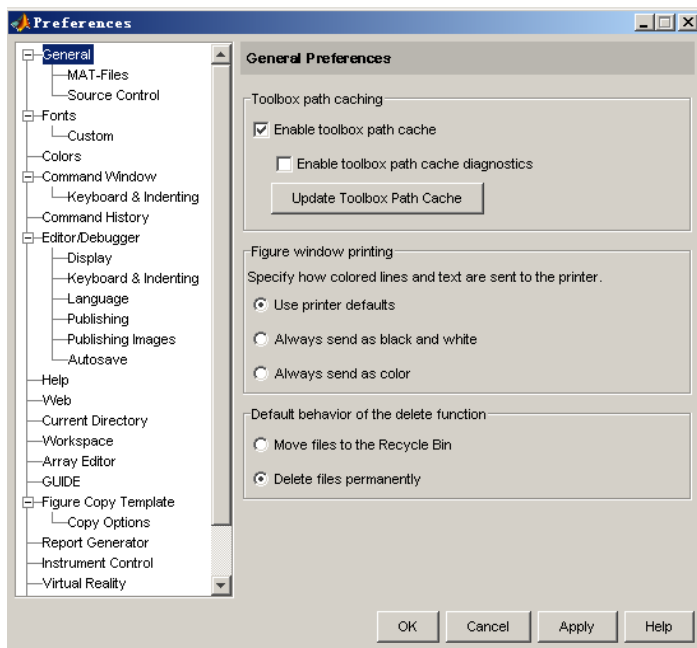


图 1.15 “Preferences” 窗口

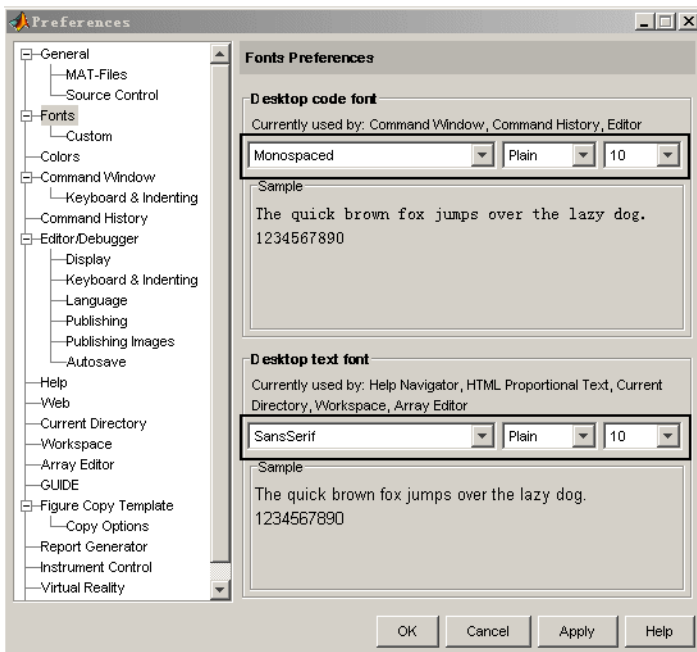


图 1.16 “Preferences” 窗口的 Fonts 设置

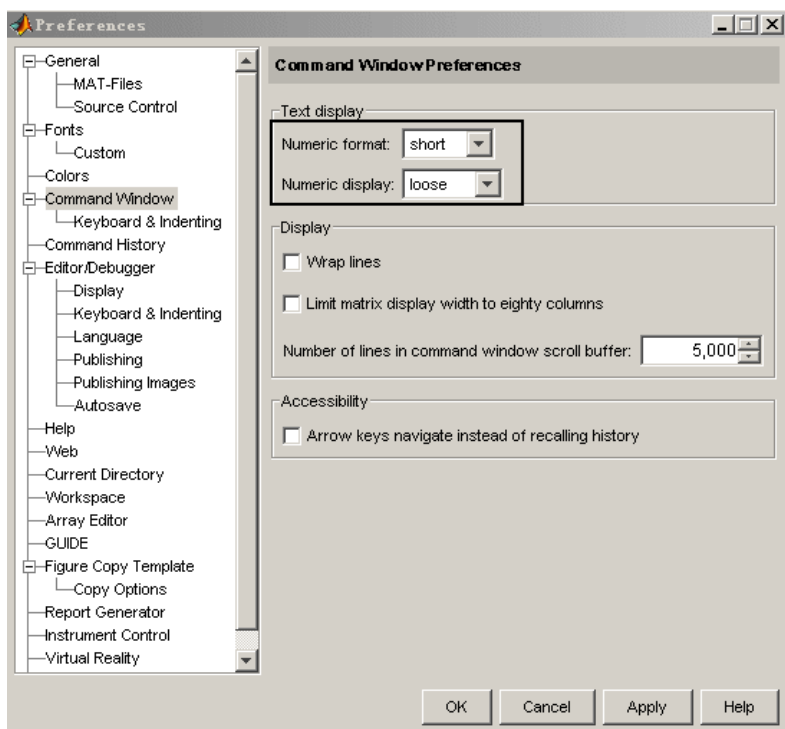


图 1.17 “Preferences” 窗口命令窗格输出设置

- Page Setup: 用于打印页面设置, 包括字体、页眉、页面布局设置。
- Print: 该菜单项用于执行打印文档任务。
- Exit MATLAB: 退出 MATLAB。

2. Edit 子菜单

Edit 子菜单主要用于常见文档的撤销、复制、粘贴、删除、查找等操作, 同时用于清除命令窗格、历史窗格、工作空间内容。子菜单界面如图 1.18 所示。其中。

- Undo: 用于撤销上一步的操作。
- Redo: 重复上一步的操作。
- Cut: 剪切选中的部分。
- Copy: 复制选中的部分。
- Paste: 粘贴已复制的内容到光标所在的位置。
- Paste Special: 打开数据导入平台, 可设置一定导入方式把剪贴板中的内容导入 MATLAB 工作空间中。
- Select All: 选中当前激活窗口中的所有内容, 用于下一步的复制操作。
- Delete: 删除选中的文件。
- Find: 可用于查找命令窗格、历史窗格内的指定内容。
- Find Files: 用于查找 MATLAB 文件, 搜索范围可设置在当前目录、MATLAB 安装路径下、或电脑中各盘符下。
- Clear Command Window: 用于清除当前命令窗格中的所有内容。
- Clear Command History: 用于清除历史窗格中的所有内容。
- Clear Workspace: 用于清除工作空间组中的所有变量。

Undo	Ctrl+Z
Redo	
Cut	Ctrl+W
Copy	Alt+W
Paste	Ctrl+Y
Paste Special...	
Select All	
Delete	Ctrl+D
Find...	
Find Files...	
Clear Command Window	
Clear Command History	
Clear Workspace	

图 1.18 Edit 菜单

3. Debug 子菜单

Debug 子菜单用于程序调试，如图 1.19 所示，主要包括调试断点设置、调试步骤的步进。具体的程序调试操作将在后续章节详细介绍。

✓ Open M-Files when Debugging	
Step	F10
Step In	F11
Step Out	Shift+F11
Continue	F5
Clear Breakpoints in All Files	
Stop if Errors/Warnings...	
Exit Debug Mode	

图 1.19 Debug 菜单

4. Desktop 子菜单

Desktop 子菜单用于桌面窗口显示控制，如图 1.20 所示。其中：

- 菜单中的第一项用于使当前激活窗格独立出 MATLAB 软件，图 1.20 中显示为“Unlucky Current Directory”，单击此菜单项可使当前路径目录浏览窗格成为独立窗口。
- “Desktop Layout”菜单项用于设置软件窗格的显示。“Default”菜单项用于恢复窗格显示至默认状态，“Command Window Only”菜单项仅显示命令窗格，“History and Command Window”菜单项显示历史和命令窗格，“All Tabbed”菜单项用于显示所有窗格，MATLAB 可根据使用者的需求设置任意组合的窗格显示方式。
- “Save Layout”菜单项用于保存用户的窗格显示模式，窗格显示方式改变后，可直接通过保存的显示模式恢复。
- “Command Window”、“Command History”、“Command Directory”、“Workspace”、“Help”、“Profiler”菜单项分别用于控制相应窗格的显示，在菜单项前有“✓”符号，表示此窗格已打开。
- “Toolbar”、“Shortcuts Toolbar”、“Current Directory Toolbar”、“Title”分别用于相应各工具条、标题栏的显示，菜单项前有“✓”符号，表示此工具条、标题栏已打开。

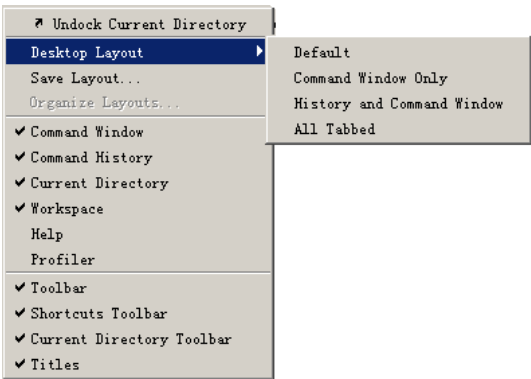


图 1.20 Desktop 菜单

5. Window 子菜单

Window 子菜单用于激活已显示的窗格为当前的活动窗格，各窗格的激活除了可以通过 Window 各菜单项外还可以通过相应的快捷键，如图 1.21 所示。

Close All Documents		
0	Command Window	Ctrl+0
1	Command History	Ctrl+1
2	Current Directory	Ctrl+2
3	Workspace	Ctrl+3

图 1.21 Window 菜单

6. Help 子菜单

Help 子菜单用于获取 MATLAB 7.0 帮助信息，如图 1.22 所示，MATLAB 帮助系统主要包括软件自带的帮助文件和网络在线帮助文档，具体的帮助系统的使用将在本章 1.4 节中详细介绍。下面介绍各菜单项的功能。

- Full Product Family Help: 用于打开 MATLAB 产品所有的帮助文件，从帮助文档最开始显示。
- MATLAB Help: 打开 MATLAB 的帮助文件，从 MATLAB 软件操作帮助文档开始，同时该菜单项可使用快捷键“F1”打开。
- Using the Desktop : 打开 MATLAB 的帮助文件，并从 Desktop 帮助文档开始。
- Using the Command Window: 打开 MATLAB 的帮助文件，并从 Using the Command Window 开始显示帮助文件。
- Web Source : 用于获取网络上的 MATLAB 帮助文档。该菜单项包括自动链接到 MathWorks 网站上的 The MathWorks Web Site 子菜单项，分别链接到相应网站产品信息 Products、出版商成员 Membership、技术支持 Technical Support KnowledgeBase、MATLAB Central、MATLAB File Exchange、MATLAB Newsgroup Access 和 MATLAB Newsletters 的子菜单项。
- Check for Updates: 用于检查网上软件的更新情况。
- Demos: 打开演示文件。MATLAB 的基础知识，各工具箱、Simulink 的使用 MATLAB 均配有相应的演示文件，观看演示文件是最快掌握 MATLAB 使用的重要途径。
- About MATLAB: 打开本安装版本的 MATLAB 序列号说明。

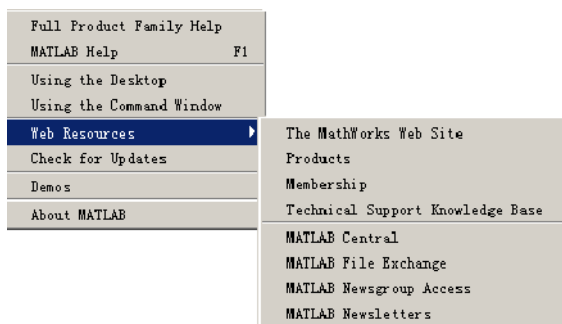


图 1.22 Help 菜单

7. View 菜单项

当用户当前活动窗口为 Current Directory、Workspace、Array Editor，主菜单将会增加 View 菜单项，同时不同窗口下的 View 菜单项是不同的，下面具体介绍不同窗口下的动态变化的 View 菜单项。

(1) “Current Directory” 窗口下的 View 菜单。

当前目录窗口下的 View 菜单如图 1.23 所示，主要用于设置当前路径下的显示文件类型。其中：

- Directory Reports: 分析当前目录下各类型 MATLAB 文件，产生分析文件相关信息的报告。
- M-Files: 当前目录下仅显示 M 类型 MATLAB 文件。
- MAT-Files: 当前目录下仅显示 MAT 类型 MATLAB 文件。
- MEX-Files: 当前目录下仅显示 MEX 类型 MATLAB 文件。
- FIG-Files: 当前目录下仅显示 FIG 类型 MATLAB 文件。
- P-Files: 当前目录下仅显示 P 类型 MATLAB 文件。
- Models: 当前目录下仅显示 Models 类型 MATLAB 文件。
- Stateflow Files: 当前目录下仅显示 Stateflow Files 类型 MATLAB 文件。
- Real-Time Workshop Files: 当前目录下仅显示 Real-Time Workshop Files 类型 MATLAB 文件。
- All MATLAB Files: 当前目录下显示所有类型 MATLAB 文件。
- All Files: 当前目录下显示所有文件。
- Folders: 当前目录下仅显示文件夹。

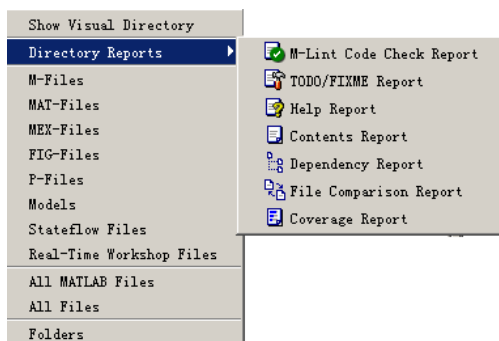


图 1.23 “Current Directory” 窗口下的 View 菜单

(2) “Workspace” 窗口下的 View 菜单。

工作空间窗口下的 View 菜单如图 1.24 所示，主要用于设置工作空间窗口数据显示。

其中：

- Choose Columns: 用于控制工作空间窗格下的数据显示，可以显示数据名称、值、维数大小、字节大小、类型。
- Sort By: 用于对工作空间窗口下的数据进行排序，包括按数据名称、值、维数大小、字节大小、类型，排序方式有升序和降序方式。

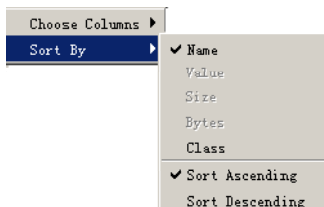


图 1.24 “Workspace” 窗口下的 View 菜单

(3) “Array Editor” 窗口下的 View 菜单。

数组编辑窗口下的 View 菜单如图 1.25 所示。

- Numeric Array Format: 用于数据精度显示控制，包括 short、shortE、long 等常见的数据类型。
- Go Up One Level: 返回上一级的数据，使结构数组等具有多级数据返回上一级数据显示。

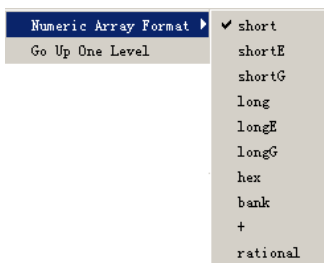


图 1.25 “Array Editor” 窗口下的 View 菜单

8. Graphics 菜单项

当用户当前活动窗格为 Workspace，主菜单将会增加 Graphics 菜单项，主要用于对工作空间窗格内的数据执行快速制图操作，如图 1.26 所示。

- New Figure: 用于打开新的绘图窗口。选择“Graphics”→“New Figure”命令打开如图 1.27 所示的“Figure”窗口，在此窗口内可以绘制新的图，之前的绘图窗口的图仍能保留，不会被覆盖。“Figure”窗口主要由菜单栏、工具栏和绘图区域组成。
- Plot Tools: 用于打开图片编辑窗口。选择“Graphics”→“Plot Tools”命令打开如图 1.28 所示的图片编辑窗口，主要包括图片布局设置、工作空间变量显示、图片属性设置、图片浏览等功能。
- More Plots: 用于设置不同类型图形绘制。选择“Graphics”→“More Plots”命令打开如图 1.29 所示的“Plot Catalog”窗口，在 Plot Variables 窗口中输入已存在于 MATLAB 工作空间内的绘图数据变量名，在 Categories 区域选择图形分类，在 Plot Types 区域选择所需绘制图形的类型，Description 区域为对所选择的图形的描述。

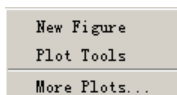


图 1.26 “Workspace” 窗口下的 Graphics 菜单

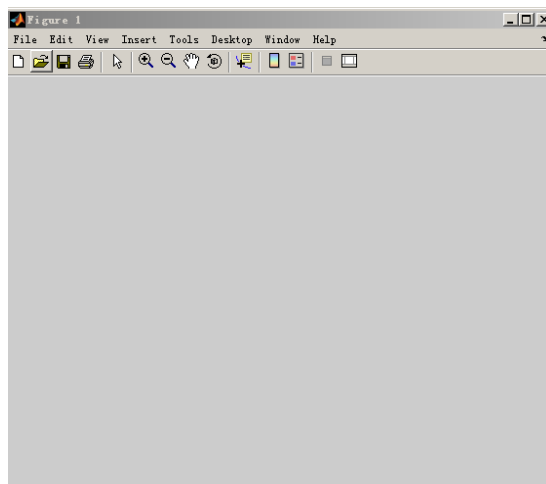


图 1.27 “Figure” 窗口

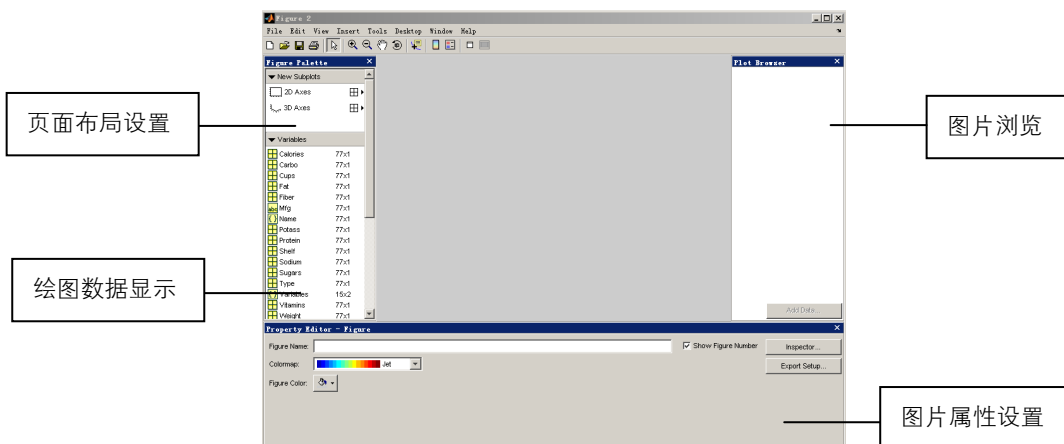


图 1.28 图片编辑窗口

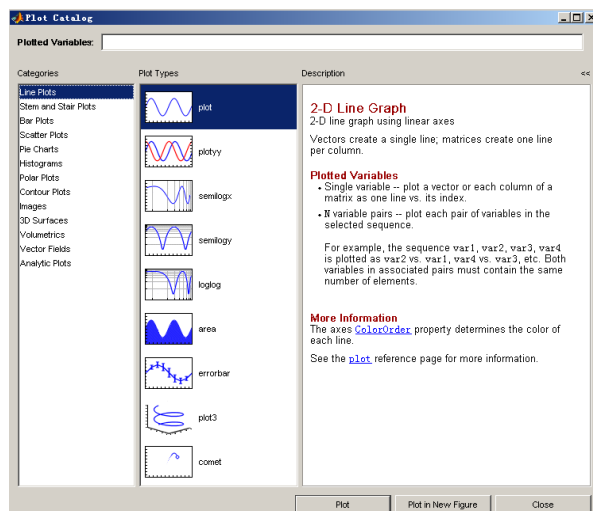


图 1.29 “Plot Catalog” 窗口

1.3.3 标题栏

MATLAB 7.0 主界面的标题栏如图 1.30 所示, 包括常见的文档操作: 新建、打开、剪切、复制、粘贴、撤销、恢复。用于打开 Smulink 窗口; 用于打开 GUIDE 窗口; 用于打开帮助文档; 用于当前目录的设置。



图 1.30 标题栏

1.3.4 命令窗口

命令窗口主要用于运行 MATLAB 函数、m 文件和显示计算结果。

1. 命令窗口的打开

当启动 MATLAB 7.0 后, 默认情况下命令窗口已自动在主界面中显示, 如果没有打开, 可选择 “Desktop” → “Command Window” 命令打开命令窗口。

单击 MATLAB 7.0 主界面 “Command Window” 窗口右上角的按钮, 将使命令窗口独立出来, 如图 1.31 所示。单击 “Command Window” 窗口右上角的按钮, 使命令窗口重新固定在 MATLAB 7.0 主界面内。以下各窗口的固定与独立都可以使用这两个按钮, 后续章节不再详细讲述。

2. 命令窗口的组成

命令窗口的菜单项与主界面的菜单项基本相同, 在此不具体介绍, 相关内容可以参考主界面菜单项的详细介绍, 同时以下各窗口中的菜单项与主界面类似的都不详细展开叙述。命令窗口内第一行显示: “To get started, select MATLAB Help or Demos from the Help menu.”, 提示用户在使用命令窗口前, 可以先阅读 MATLAB 的帮助文档和演示程序, 如图 1.30 所示。在提示信息后为符号 “>>”, 从此处开始输入需要运行的表达式、函数等。

单击鼠标右键可打开如图 1.32 所示的快捷菜单。其中, Evaluate Selection 用于运行选中的函数、表达式; Open Selection 用于打开选中函数的 m 文件; Help on Selection 用于打开选中函数的帮助文件。

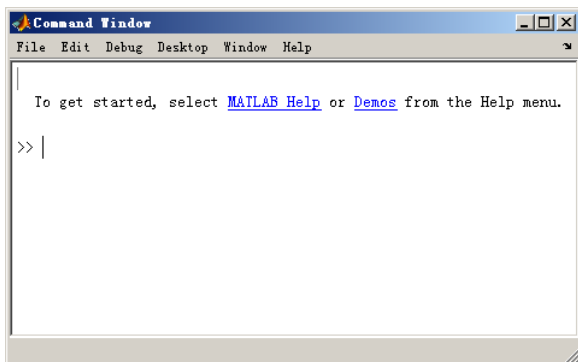


图 1.31 “Command Window” 窗口

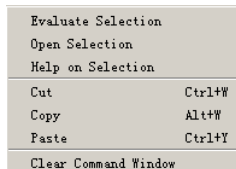


图 1.32 “Command Window” 窗口的快捷菜单

3. 命令窗口的操作

命令窗口的操作主要包括函数、程序运行和数据的输入。命令窗口操作规定:

- 在 “>>” 符号后输入相应的代码后按 “Enter” 键, 可完成代码的执行。
- 执行的代码不需要显示中间过程, 加上分号, 可提高代码的执行效率。

- 一行可输入多行代码，中间用“,”或“;”分隔。
- 同时运行输入的多行代码：输入完一行代码后，按“Shift+ Enter”键转入下一行代码的编辑，暂时不运行输入的代码，待多行代码输入完毕后，按“Enter”键正常运行所有输入代码。
- 命令行中有多余空格不影响代码的执行，但为便于阅读，尽量应减少不必要的空格。
- Tab 自动输入函数完整的函数名和完整的变量名，前提是函数已存在于 MATLAB 搜索路径或当前目录，变量已存在于工作空间内。
- 数据输出显示形式和精度的控制参见 1.3.2 节主菜单中 File 子菜单, Preference 设置的相关介绍。

命令窗口常用命令介绍如下。

(1) 清除命令如下。

- clear: 清除工作空间内的所有变量。
- clear all: 从工作空间清除所有变量和函数。
- clear name1 name2 name3...: 从工作空间内清除名为 name1、name2、name3 的变量。
- clear global: 从工作空间清除所有全局变量。
- clc: 清除命令窗口中的所有内容。
- clf: 清除当前图形窗口中的所有图形。
- cla: 清除当前图形窗口中坐标轴内的内容，保留坐标轴。

(2) 数据导入命令如下。

- load filename: 导入 filename.mat 数据文件中的所有变量。
- load filename x: 仅导入 filename.mat 数据文件中的变量 x。
- load filename x y z ...: 导入 filename.mat 数据文件中的变量 x、y、z。

(3) 保存命令如下。

- save: 以数据文件 matlab.mat 保存工作空间内的所有变量。
- save filename: 保存工作空间内的所有变量于数据文件 filename.mat 中。
- save filename v1 v2...: 在数据文件 filename.mat 中保存变量 v1、v2 等。
- save filename v1 v2 append: 把变量 v1、v2 添加到已有数据文件 filename.mat 中保存。

(4) 关闭命令如下。

- close: 关闭当前图形窗口。
- close(h): 关闭图像句柄 h 的窗口。
- close all: 关闭所有的图形窗口。

(5) 查看命令如下。

- who: 查看当前工作空间内存储的变量名。
- whos: 查看当前工作空间内存储的变量详细信息，包括变量名、变量大小、变量数据类型等。
- which fun: 查看函数 fun 存储路径。
- what: 查看当前路径下的所有 MATLAB 类型文件。
- what dirname: 查看指定路径 dirname 下的所有 MATLAB 类型文件。

(6) 路径相关命令如下。

- cd: 显示当前路径。
- cd('directory'): 设置 directory 为当前路径。
- cd ..: 返回当前目录的上一层次目录。

【例 1.1】命令窗口的操作示例。

(1) 输入变量，在命令窗口输入如下代码：

```
>>A=[1 2 3;4 5 6;7 8 9;10 11 12]
```

窗口中将会显示输入的数据 A，同时数据在 MATLAB 工作空间内将以矩阵的形式存在。

```
A =  
    1     2     3  
    4     5     6  
    7     8     9  
   10    11    12
```

(2) 运行函数，在命令窗口输入如下代码：

```
>> ones(3)
```

MATLAB 返回结果：

```
ans =  
    1     1     1  
    1     1     1  
    1     1     1
```

4. 命令窗口的编辑

命令窗口提供键盘操作用于重新编辑已执行的命令，当输入代码出错后无须重新输入代码，仅利用键盘“↑”键，即可完成操作。例如在命令窗口输入如下代码：

```
y=sin(30)
```

由于函数“sin()”在输入的时候拼写错误，出错后，此时无须重新写此命令，只要按“↑”键，即可调出已执行的错误代码，重新编辑即可。






1.3.5 当前目录浏览窗口

当前目录浏览窗口用于显示当前目录下的文件，包括文件名、文件类型、文件上次修改时间、描述信息的显示。可快速地对当前路径下的文件执行一般的 Windows 操作，打开、删除、剪切、重命名等，同时可在 MATLAB 环境下打开文件。

1. 当前目录浏览窗口的显示

选择“Desktop”→“Current Directory”命令，打开当前目录浏览窗口，独立出当前目录浏览窗口，如图 1.33 所示。

2. 当前目录浏览窗口的组成

当前目录浏览窗口由菜单栏、当前路径框、工具栏图标和当前目录显示组成。其中工具栏图标用于返回上一级文件目录；图标用于在当前目录新建文件夹；图标用于在当前目录中查找文件；图标用于以可视化的方式查看文件；图标用于产生不同类型文件的分析文档。

单击鼠标右键可打开如图 1.34 所示的快捷菜单。其中，

- Open：用于打开文件，如果为 m 文件，将在代码编辑窗口显示代码。
- Run：可直接运行 m 文件。
- View Help：用于查看 MATLAB 帮助文档。
- Open as Text：以文本的形式打开文件。
- Open Outside MATLAB：以非 MATLAB 打开方式打开文件，按照文件默认的打开方式打开。
- Import Data：通过数据导入平台导入数据。
- File Filter：设置当前路径下的显示文件类型。

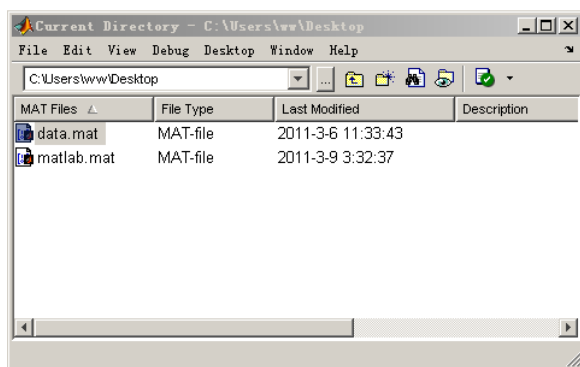


图 1.33 “Current Directory” 窗口

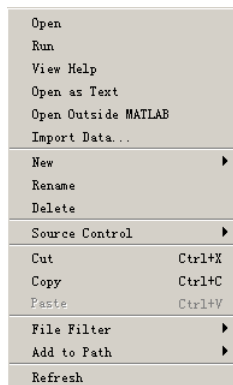


图 1.34 “Current Directory” 窗口的快捷菜单

3. 当前目录浏览窗口的操作

当前目录浏览窗口的操作主要包括常规的文档操作（打开、删除等）和 MATLAB 软件特有的一些操作（文档分析、可视化模式显示等），下面以一个例子详细介绍当前目录浏览窗口的操作。

【例 1.2】当前目录浏览窗口的操作，在本例中将完成当前目录设置、数据导入、文件打开、文件查找等的操作。

（1）设置当前目录：在当前目录对话框 中，选择当前目录为 C:\MATLAB701\work，当前目录浏览窗口内将显示该目录下的文件，如图 1.35 所示，此处显示了该目录下所有类型的文件。

（2）数据导入：当前目录浏览窗口内数据 “fisheriris.mat” 的导入方法如下，选中数据文件 fisheriris.mat，单击鼠标右键，在弹出的快捷菜单内选择 “Import Data” 命令，如图 1.36 所示，将通过数据导入平台导入数据。

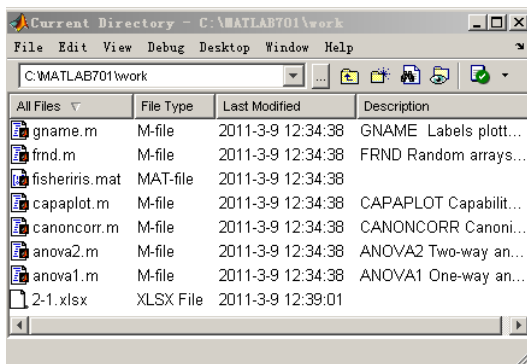


图 1.35 设置当前路径

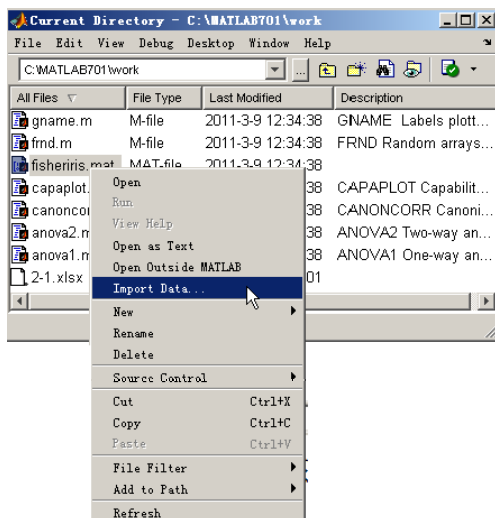




图 1.36 “Current Directory” 窗口数据导入

（3）文件打开：对于当前目录下的 m 文件，选中 “find.m” 文件并单击鼠标右键，在弹出的快捷菜单中选择 “Open” 命令，或者直接双击文件，将在代码编辑窗口中打开函数 find 的文件；而非 MATLAB 文件 “2-1.xlsx”，可选择快捷菜单中的 “Open Outside MATLAB” 方式打开，文件将会以 Excel 方式打开。

(4) 文件查找: 单击标题栏图标, 打开文件查找窗口, 可以实现文件搜索和文件内包含内容的搜索。在图 1.37 所示的查找窗口中, 在 Find files named 列表框中写入需要查询的文件名“anova1”, 在 Find files containing text 列表框中写入需要查询的内容“anova”。Lock in 下拉列表框用于设置搜索文件的范围, 本例中选择在 Current directory 当前路径下搜索。Include subdirectories 复选框用于选择搜索是否在搜索路径下的子文件下进行。Advanced Options 用于文件搜索的高级参数设置。

单击“Find”按钮, 执行文件查找命令, 在窗口右半部分将显示文件查找的结果, 包括查找到的文件名、查找关键内容所在的行数、查找关键内容所在行的代码。

(5) 可视化显示: 标题栏图标用于以可视化的方式查看文件。如图 1.38 所示, 以可视化的方式显示当前路径下的文件。单击文件名, 可直接打开文件, 在此模式下原来当前目录浏览窗口下的操作都只需单击相应的蓝色链接实现。

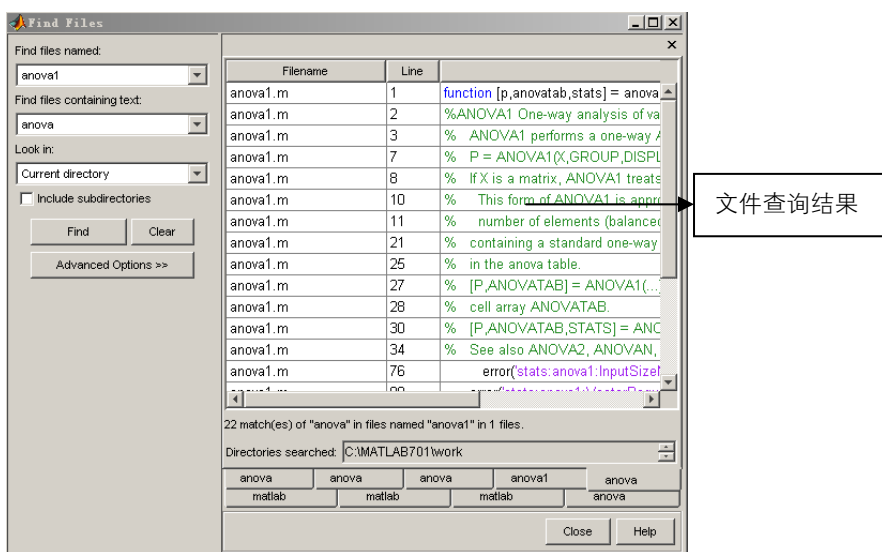


图 1.37 文件查找窗口

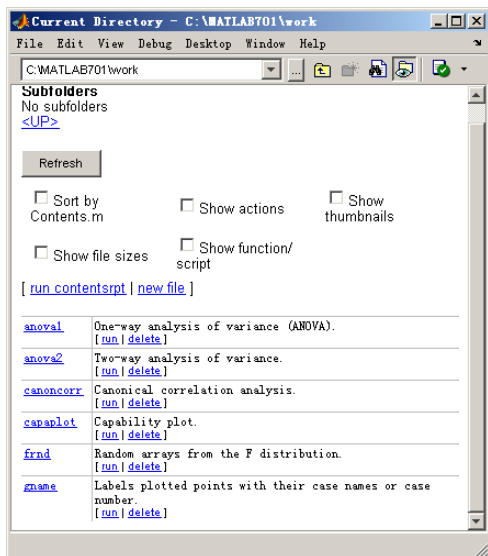



图 1.38 “Current Directory” 窗口可视化显示

(6) MATLAB 文件分析报告的生成: 选标题栏图标的下拉列表中的选项, 可用于生成不同类型的 MATLAB 文件的分析报告, 本例中选择“M-link Code Check Report”选项, 将产生如图 1.39 所示的 m 文件分析报告。

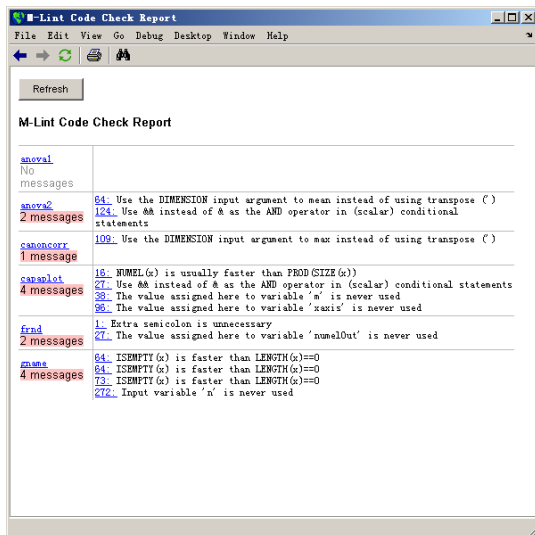








图 1.39 m 文件分析报告

1.3.6 工作空间浏览窗口

工作空间浏览窗口主要用于对 MATLAB 内存中的数据进行查看、编辑等操作。工作空间浏览窗口的打开与上述菜单的显示、打开方法类似, 选择“Desktop”→“Workspace”命令。工作空间浏览窗口主要显示变量的名称、数据结构、字节大小、数据类型, 如图 1.40 所示。各菜单功能如下:

- 按钮用于在工作空间内的新建变量。
- 按钮用于打开数组编辑窗口, 查看选中的变量。
- 按钮用于从外部文件导入数据; 按钮用于保存选中的变量; 按钮用于删除相关变量。
- 按钮用于对选中的变量快速作图。

同时工作空间浏览窗口提供如图 1.41 所示的快捷菜单, 可用于当前变量的打开、保存、复制、建立副本、删除、重命名、编辑、各类型图的绘制。

工作空间的变量在退出 MATLAB 后会清空, 如果希望在下次使用 MATLAB 时继续使用本次工作空间内建立的变量, 则需要在硬盘空间存储。

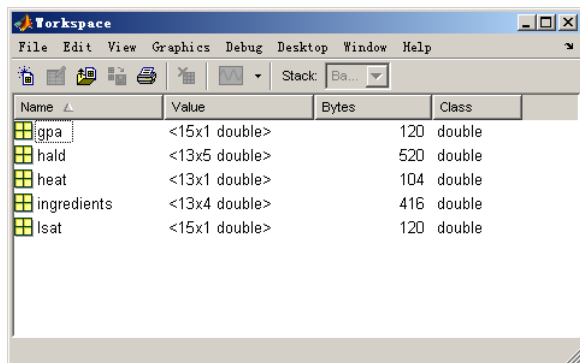


图 1.40 “Workspace” 窗口

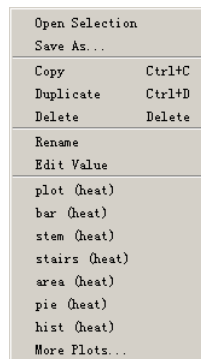



图 1.41 “Workspace” 窗口的快捷菜单

【例 1.3】工作空间浏览窗口操作，在本例中将进行从数据导入、数据显示到数据保存等一系列数据工作空间窗口操作。

(1) 导入数据：MATLAB 中数据导入有多种方式，可以通过数据导入平台导入、命令行导入等方式，在本例中主要介绍如何在“Workspace”窗口导入数据。单击图标选择从外部文件导入数据，选择导入数据文件，弹出如图 1.42 所示的“Import Wizard”窗口。单击“Finish”按钮完成数据导入工作，在数据工作空间将建立变量 meas 和 species。

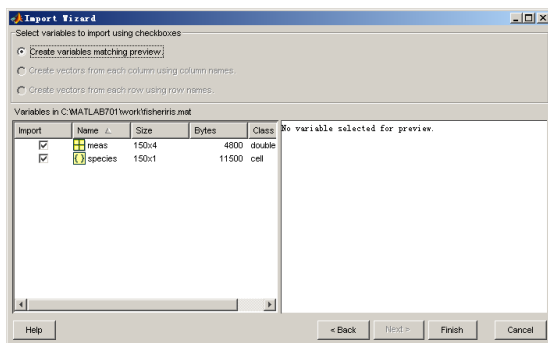


图 1.42 “Import Wizard” 窗口

(2) 数据查看：双击选中的变量可以打开如图 1.43 所示的数组编辑窗口，在此窗口内可以查看变量的具体数值。选中 meas 变量所有数据并单击鼠标右键，弹出如图 1.44 所示的快捷菜单。选择“Plot selected columns”命令，将绘制出如图 1.45 所示的折线图，用于显示 meas4 列数据的变化趋势。同时数组编辑窗口还提供对数据的剪切、复制、粘贴、插入、删除、清除等数组编辑功能。

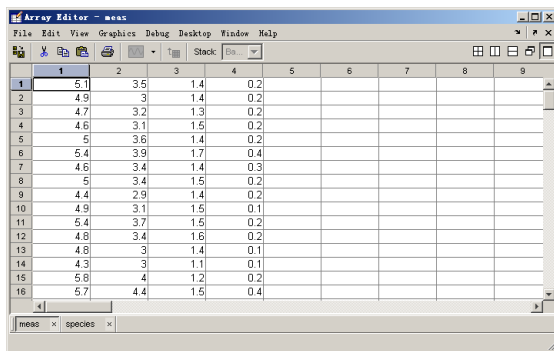


图 1.43 “Array Editor” 窗口

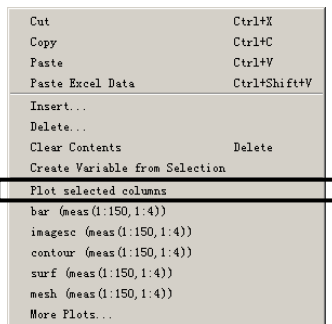


图 1.44 “Array Editor” 窗口的快捷菜单

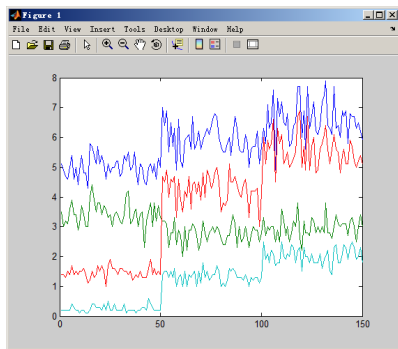




图 1.45 “Array Editor” 窗口数据的图形显示

(3) 数据保存: 选中需要保存的数据 meas, 单击按钮 , 或在右键快捷菜单中选择 “save as” 命令, 保存 mat 格式的数据。

(4) 数据删除: 选中需要删除的数据 meas, 单击按钮 , 或在右键快捷菜单中选择 “delete” 命令, 删除变量。如果工作空间内的数据需要全部删除, 可以在命令窗口中输入 “clear” 命令。

1.3.7 历史命令窗口

历史命令窗口主要记录近期在命令窗口运行过的代码, 包括命令、函数、表达式等历史运行信息。可以在历史命令窗口进行历史命令的查找、粘贴、复制与重运行等操作。

1. 历史命令窗口的显示

当启动 MATLAB 7.0 后, 默认情况下历史命令窗口已自动在主界面中显示, 如果没有打开, 可选择 “Desktop” → “Command Directory” 命令, 打开如图 1.46 所示的历史命令窗口。

2. 历史命令窗口的操作

历史命令窗口中与上述其他窗口类似的查找、剪切、复制等操作不再展开叙述, 这里主要讨论历史窗口的特殊操作, 主要在如图 1.47 所示的快捷菜单中完成。

(1) 历史命令的重运行: 在历史命令窗口中选中需要重运行的历史命令, 双击或者单击鼠标右键, 在弹出的快捷菜单中选择 “Evaluate Selection” 命令, 在命令窗口中将运行选中的命令。

(2) 历史命令 m 文件的重新生成: 选中历史命令窗口中的命令, 在快捷菜单中选择 “Create M-File” 命令可以生成独立的 m 文件。

(3) 历史命令代码分析: 选择快捷菜单中的 “Profile Code” 命令对选中的代码进行分析。

(4) 历史命令窗口命令的删除: 提供了窗口删除选中的命令、删除命令至选中的命令、删除所有命令 3 种删除方式, 上述功能的实现分别对应快捷菜单中的 “Delete Selection”、“Delete to Selection”、“Clear Entire History” 命令。

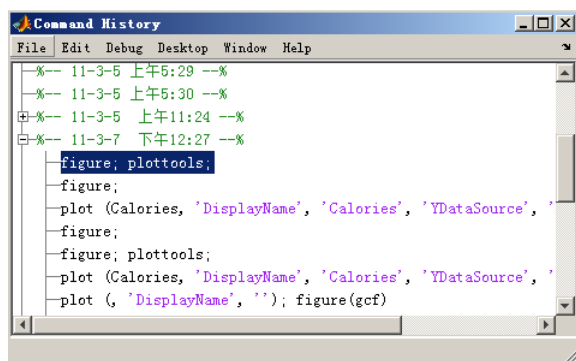


图 1.46 “Command Directory” 窗口

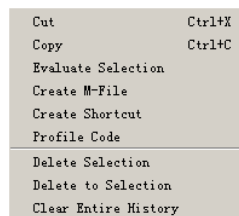


图 1.47 “Command Directory” 窗口的快捷菜单

1.4 帮助系统

作为一个优秀的软件, MATLAB 为广大用户提供了有效的帮助系统。MATLAB 7.0 版本为用户提供了联机帮助系统、远程帮助系统、演示程序、命令查询系统等多种方式帮助, 这些无论对于入门读者还是经常使用 MATLAB 的人员都是十分有用的, 经常查阅 MATLAB 帮助文档, 可以帮助我们更好地掌握 MATLAB。

获得帮助的主要工具是帮助浏览器, 它提供了所有已安装产品的帮助文档, 以帮助使用者全


面了解 MATLAB 功能。如果 Internet 连接可用，可以观看在线帮助和功能演示的视频。

1.4.1 帮助浏览器

帮助浏览器是整合 html 形式的帮助文档在 MATLAB 桌面环境中，安装 MATLAB 软件时会自动安装所安装产品的帮助文件和演示程序。

1. 帮助浏览器的打开

帮助浏览器的打开主要有以下几种方式：

- 单击 MATLAB 主界面的标题栏中的按钮 。
- 选择主界面 “Help” → “Full Product Family Help” 命令，或 “Help” → “MATLAB Help”、“Help” → “Using the Desktop”、“Help” → “Using the Command Window” 命令。
- 在 MATLAB 各个界面中按快捷键 “F1”。
- 在命令窗口运行 “doc” 命令或者在利用命令查询系统获取某函数的帮助信息后，直接在其后给出的相关参考链接 “doc fun” 进入函数的帮助浏览器窗口。

2. 帮助浏览器窗口的组成

选择上述打开方式的任意一种打开帮助浏览器，如图 1.48 所示，帮助浏览器的左半面显示帮助文档导航器，方便用户查找相应内容的帮助文档，可按主题、索引、搜索、演示文档分类查找相应内容的帮助文件。右半面区域为帮助文档的显示。

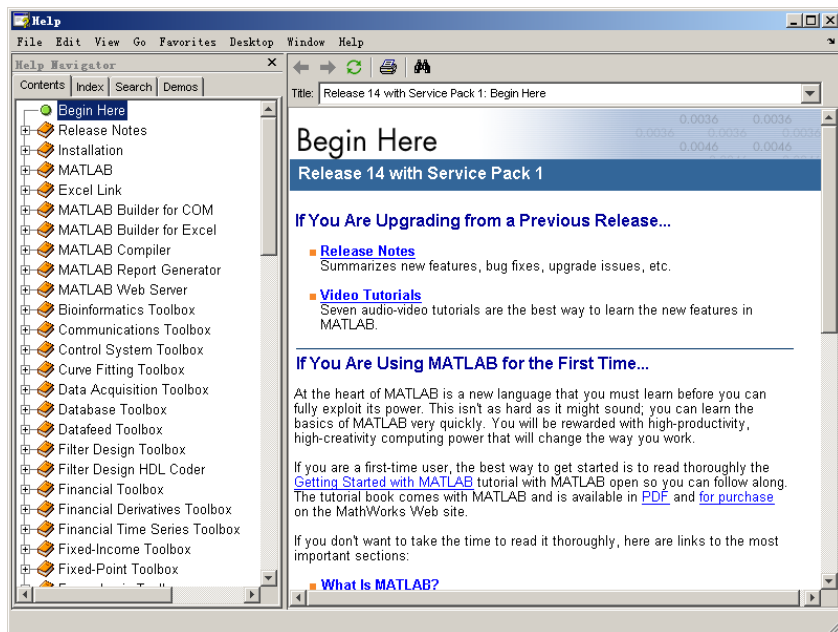












图 1.48 帮助浏览器

3. 帮助浏览器窗口的使用

1) 帮助文档的查找

帮助浏览器导航器提供了主题界面、索引界面、搜索界面、演示文档界面，用于查找相应内容的帮助文档。其中：





- 按内容分类主题界面是节点可展开的树形目录，为用户提供 MATLAB 版本信息、软件安装、基础知识、主要工具箱使用的帮助文档，用户可以通过单击按内容分类的节点，打开相应

内容的帮助文档。其中一级主题目录相应图标分别对应的帮助文档如下：目录图标为软件快速入门，目录图标为 MATLAB 帮助文档，目录图标为 Simulink 产品帮助文档；二级主题目录相应图标分别对应的帮助文档如下：图标为快速入门指导，图标为实例和演示程序，图标为主要功能的实现说明，图标为函数说明，图标为版本信息说明，图标为获取帮助文档的 PDF 版本，图标为远程帮助。

- 按函数首字母建立索引界面，用户可以根据需要查找内容的关键词，在索引窗口打开相关内容的帮助文档。
- 搜索界面提供了搜索对话框，可在其中搜索包含指定内容的帮助文档。
- 演示文档界面按 MATLAB 主程序、工具箱、Simulink、模型块 4 部分内容分类浏览相应内容的演示程序，同时单击演示文档界面树形节点可打开其中的子内容，选择需要查找的演示文档。

2) 帮助文档的阅读

在通过帮助浏览器导航窗口查找到相应内容的帮助信息后，在右侧区域将显示相应的帮助文档。帮助文档包括函数、各功能实现、程序演示、版本信息、在线帮助链接等部分。

帮助文档上方提供了标题栏和 Title 下拉列表框以便于阅读帮助文档，如图 1.48 所示。其中，标题栏中的按钮和分别用于阅读页面的后退和前进，按钮用于刷新当前阅读界面，按钮用于查找帮助内容。Title 下拉列表框可用于选择打开之前阅读过指定 Title 的帮助界面。

【例 1.4】以帮助主题中 MATLAB 子模块为例，演示帮助文档的组成与阅读。

单击帮助导航器 Help Navigator 中 Content 主题界面中的 MATLAB 目录项，打开如图 1.49 所示的 MATLAB 产品帮助文档，该部分帮助文档主要介绍 MATLAB 主程序设计的基本功能和相应函数的使用。其中：

- Functions 部分提供按不同功能（By Category）和首字母顺序（Alphabetical List）分类两种方式浏览函数帮助文档。按功能分主要分为桌面环境函数、矩阵运算、程序设计和数据类型、文件输入/输出、绘图函数、三维可视化函数、GUIDE 函数和外部应用程序接口函数。
- Handle Graphics：提供图形对象设置的帮助文档，提供坐标轴等图形对象设置的帮助文档。
- Documentation Set：显示了快速入门、用户指南、程序设计技巧、实例文件等部分的帮助文档快速链接。其中，Getting Started 帮助文档可使读者快速入门，掌握 MATLAB 基本操作，Use Guides 部分包括 MATLAB 基本操作各主要功能的帮助文档，Programming Tips 部分为程序设计相关内容帮助文档，例如变量的命名、程序的调试等内容。
- Examples in Documentation：为相关主题所有帮助文档的实例，通过实例的学习可以很好地学习 MATLAB 各功能的使用，同时能提高 MATLAB 编程能力。帮助窗口内的所有实例，可以通过选中相关内容，通过帮助浏览器中的右键弹出式快捷菜单，执行“Evaluate Selection”命令，在命令窗口将直接运行所选程序。
- Product Demos：用于显示演示程序。
- What's New：用于显示当前所安装 MATLAB 版本的信息，主要为当前版本的更新信息。
- Printing the Documentation Set：用于打印保存帮助文档的 PDF 版本。
- The MathWorks Web Site Resources：提供了链接，直接打开 MathWorks 公司主页上的相关帮助文档的网页。

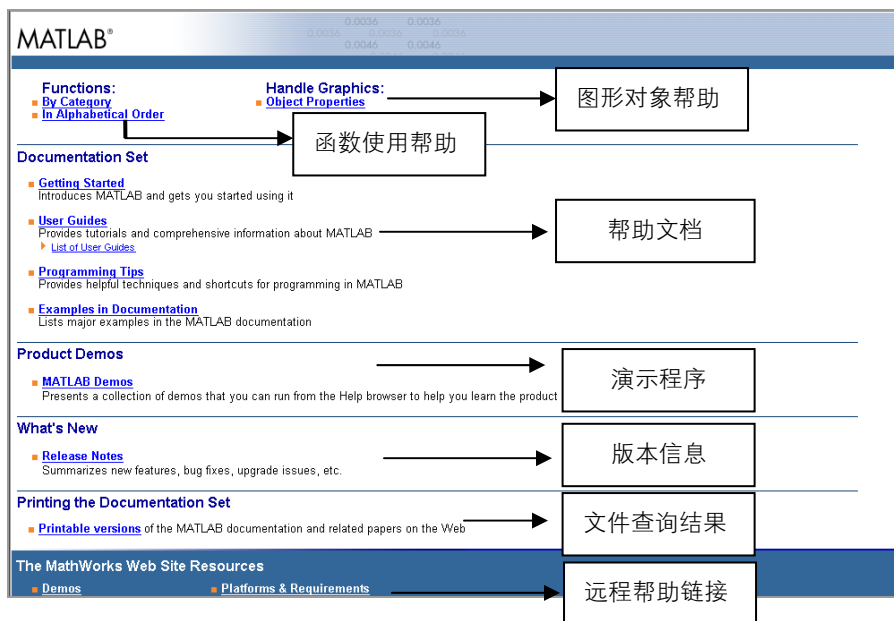


图 1.49 MATLAB 主程序的帮助内容

1.4.2 命令帮助系统

命令帮助系统提供在命令窗口中输入帮助命令来获取相关函数或软件的帮助信息。命令帮助系统是获取指定函数帮助信息的最为便捷的途径，主要提供的帮助信息为相应程序 m 文件中的帮助信息，同时在命令窗口中获取的帮助信息包含与帮助浏览器相应内容的链接，可以进一步查看更为完整的帮助信息。经常在命令窗口中查阅函数的帮助文档，对于 MATLAB 使用者是极为有益的，

命令帮助系统主要使用的函数命令为 `help` 和 `lookfor`，`help funname` 显示相关函数帮助注释区内容，`lookfor funname` 显示包含函数名的相关内容，查询条件比较宽松，只要包含 `funname` 即可。

【例 1.5】用 `help` 命令查看 `max()` 函数的帮助信息。

在命令窗口中输入命令“`help max`”，如下所示。函数帮助信息首先为函数具体用法，之后以一简单实例演示函数使用，最后给出相关函数 `min`、`median`、`mean`、`sort` 的帮助链接和 `max` 函数在帮助浏览器中帮助文档的链接。

```
>> help max
MAX    Largest component.

For vectors, MAX(X) is the largest element in X. For matrices,
MAX(X) is a row vector containing the maximum element from each
column. For N-D arrays, MAX(X) operates along the first
non-singleton dimension.
[Y,I] = MAX(X) returns the indices of the maximum values in vector I.
If the values along the first non-singleton dimension contain more
than one maximal element, the index of the first one is returned.
MAX(X,Y) returns an array the same size as X and Y with the
largest elements taken from X or Y. Either one can be a scalar.
[Y,I] = MAX(X,[],DIM) operates along the dimension DIM.
When complex, the magnitude MAX(ABS(X)) is used, and the angle
ANGLE(X) is ignored. NaN's are ignored when computing the maximum.
Example: If X = [2 8 4
                 7 3 9]
max(X,[],2) is [8
                9], and max(X,5) is [5 8 5
                                     7 5 9].
```



```

See also min, median, mean, sort.
Overloaded functions or methods (ones with the same name in other directories)
    help quantizer/max.m
    help fints/max.m
    help localpspline/max.m
    help localpoly/max.m
Reference page in Help browser
    doc max

```

1.4.3 远程帮助系统

当需要获取未安装的 MATLAB 产品的帮助文档, 或获取最新版本的帮助文档, 或希望在未安装 MATLAB 软件的电脑上阅读帮助文档, 可以在 MATLAB 网站 (<http://www.mathworks.com/access/helpdesk/help/helpdesk.shtml>) 上获取 MATLAB 所有产品的最新帮助文档。同时在 MathWorks 公司的主页 (<http://www.mathworks.com>) 上可以找到很多有用的信息, 包括产品的详细介绍、定期的网上研讨会、用户反馈、软件使用问题官方解决方案等。

1.5 如何学习 MATLAB

如何学习 MATLAB? 这是一个仁者见仁, 智者见智的问题, 在这里笔者仅结合自己使用 MATLAB 多年的经验, 与大家分享学习的心得体会, 供不同层次的读者参考。

首先, 对于零基础的读者来说, 并不建议这部分读者去阅读详细的 MATLAB 英文帮助文档, 虽然英文帮助文档是对 MATLAB 软件讲解的最权威、最全面的说明书, 但是由于广大使用 MATLAB 的用户并没有专业学习过数学, 所以对于帮助文档中很多数学上的英文词汇也不是很了解。同时由于 MATLAB 帮助文档的权威性和全面性, 因此导致各个主题的帮助文档内容过于丰富, 有几百甚至上千页的英文文档, 能够全部看完也是一个很大的挑战。因此, 对于入门读者来说建议以阅读 MATLAB 软件相关的中文书籍为主, 最好是实例比较多的教材, 便于理解。

也许对于入门读者在最初的阅读中文书籍的过程中会感到枯燥乏味, 这个时候建议边看边练。书中的代码自己实践一下, 当你自己通过软件实现函数、程序的时候, 会产生一种成就感, 有了成就感可以慢慢培养学习的兴趣。当看到原来笔算需要花费很长时间解决的问题, 现在通过软件很快、很轻易地就可以解决时, 就会对 MATLAB 的学习产生浓厚的兴趣, 激发你进一步去了解它更多的功能。

其次, 学习 MATLAB 是一个很漫长的过程。如果你之前接触过其他的编程语言, 那么在最初学习使用 MATLAB 时, 你会很自然地受到其他编程语言的影响。的确, 在最初的学习过程中会使你学得更快, 因为毕竟 MATLAB 是基于 C 语言编写的, 很多地方与一般的程序语言也很类似。但是建议在此阶段, 读者不要过多地依赖于之前学习的其他编程语言的语法规则等, 因为 MATLAB 是基于矩阵运算的, 这是其他编程语言无法相比的。很多时候你写一段 for 循环可以实现的, 其实通过矩阵运算可以更方便、更高效地实现。

如果你之前没有接触过其他的编程语言, 那么恭喜你, 你接触到了 MATLAB 这一能最快实现你想法的编程语言。MATLAB 编写程序比较简单, 读者只需掌握基本操作知识, 然后需要实现什么具体的功能只要调用现成的函数就可以了。一般的操作目的在 MATLAB 中都存在现成的函数, 函数的查找可以通过书本、网络、帮助文档的搜索。查找到相应函数后, 在 MATLAB 命令窗口中使用 “help+函数名” 可以方便地查找到函数的用法。

最后, 再给已有一定基础的 MATLAB 使用经验的读者一点建议, 阅读其他人, 特别是高手的

程序，你会学习到很多。也许很多时候你的代码和高手的代码有相同的功能，但是他们能通过很简单、有效的方法来实现，而你的代码却比较复杂。因此，定时地做好总结工作，积累一些算法、使用小技巧，才能更加熟练地使用 MATLAB 的各种功能。

1.6 本章小结

本章主要介绍了 MATLAB 软件的基本知识，对于入门读者尤为重要。通过本章的学习，读者首先对 MATLAB 软件的发展历程、优势特点、系统组成等有较为系统全面的认识；然后介绍了 MATLAB 安装、MATLAB 用户界面方面的知识。MATLAB 图形化界面使用方便，且易掌握。接着介绍了 MATLAB 强大的帮助系统，熟练掌握帮助系统的操作将有助于读者在遇到问题时能尽快解决问题；最后，笔者结合自己的经验与广大读者分享了 MATLAB 的学习使用心得体会。

通过本章的学习，希望每个读者都能对 MATLAB 的基本知识有一个系统的认识，在后续章节中将带领大家进一步体会 MATLAB 的优势、特点。

第2章

MATLAB 的数据类型

MATLAB 中有 15 种基本数据类型，其中常用的数据类型主要是整型、浮点型、逻辑类型、结构体、元胞数组及字符串等。MATLAB 数据类型在使用中与其他编程语言相比，有一个突出的特点，即不用对变量的数据类型进行定义，MATLAB 软件会自动依据变量被赋值的情况，生成相应数据类型的数据。本章将详细介绍 MATLAB 的主要数据类型及不同数据类型之间的转换。

2.1 整型

MATLAB 7.0 中的整型数据类型，按照有无符号位可以分为有符号位整数和无符号位整数，而按照数据类型的字节数又可分为 8 位、16 位、32 位、64 位的整数类型。在使用中主要根据实际存储的数据的数值大小范围来确定数据类型，在保证数据精度的同时尽量避免不必要的内存的浪费，提高运算速度。其中有符号位整型需要一个 1 位来表示数据为负，因此有符号位的整型数据 8 位、16 位、32 位、64 位的范围分别为 $-2^7 \sim 2^7 - 1$ 、 $-2^{15} \sim 2^{15} - 1$ 、 $-2^{31} \sim 2^{31} - 1$ 、 $-2^{63} \sim 2^{63} - 1$ ；无符号位的整型 8 位、16 位、32 位、64 位的范围分别为 $0 \sim 2^8 - 1$ 、 $0 \sim 2^{16} - 1$ 、 $0 \sim 2^{32} - 1$ 、 $0 \sim 2^{64} - 1$ 。

不同类型整型的生成函数如下。

- `int8(x)`: 有符号 8 位整型数据生成。
- `int16(x)`: 有符号 16 位整型数据生成。
- `int32(x)`: 有符号 32 位整型数据生成。
- `int64(x)`: 有符号 64 位整型数据生成。
- `uint8(x)`: 无符号 8 位整型数据生成。
- `uint16(x)`: 无符号 16 位整型数据生成。
- `uint32(x)`: 无符号 32 位整型数据生成。
- `uint64(x)`: 无符号 64 位整型数据生成。

【例 2.1】 不同类型整型数据的生成。

```
>> x=int8(12)           %有符号 8 位整型数据 x 生成，其值为 12
x =
    12
>> y=uint16(x)          %x 转换为无符号 16 位整型数据 y
y =
    12
>> z=int8(128)           %有符号 8 位整型数据 z 生成，128 超过最大上限，其值为 127
z =
   127
>> xx=int8(-8)           %有符号 8 位整型数据 xx 生成，其值为-8
xx =
    -8
>> yy=uint8(-8)          %无符号 8 位整型数据 yy 生成，由于负数超过其下限，yy 值为 0
```

```
yy =
    0
>> zz=int32(-8)           %有符号 32 位整型数据 zz 生成，其值为-8
zz =
    -8
>> whos                   %查看工作空间的变量
Name      Size      Bytes  Class
x         1x1         1   int8 array
xx        1x1         1   int8 array
y         1x1         2   uint16 array
YY        1x1         1   uint8 array
z         1x1         1   int8 array
zz        1x1         4   int32 array
Grand total is 6 elements using 10 bytes
>> isinteger(zz)          %判断变量 zz 数据类型是否为整型
ans =
     1
>> x=zeros(3,'uint8')    %全零矩阵的生成，并定义其数据类型为 8 位无符号整型
x =
     0     0     0
     0     0     0
     0     0     0
```

intmin()和 intmax()函数可用于确定整型数据类型的数据范围，其中 intmin()函数用于求数据类型下限，intmax()函数用于求数据类型上限。函数的调用格式如下所示。

- intmin(classname): 指导数据类型下限计算。
- intmax(classname): 指导数据类型上限计算。

【例 2.2】 不同整型数据类型数值范围计算。

```
>> intmin('int8')         %有符号 8 位整型下限计算
ans =
   -128
>> intmax('int8')         %有符号 8 位整型上限计算
ans =
    127
>> intmin('int32')        %有符号 32 位整型下限计算
ans =
-2147483648
>> intmin('uint32')       %无符号 32 位整型下限计算
ans =
     0
```

2.2 浮点型

在 MATLAB 7.0 中浮点型变量包括单精度浮点型和双精度浮点型，分别为 32 位和 64 位，使用 single()和 double()函数定义及转换。默认状态下在 MATLAB 中输入的数据即为双精度浮点型。

【例 2.3】 不同类型浮点型的生成。

```
>> x=single(32)           %单精度浮点型数据 x 生成，其值为 32
x =
    32
>> y=double(100000)       %双精度浮点型数据 y 生成，其值为 100000
y =
 100000
>> z=10                   %默认状态下数据双精度浮点型数据输入
z =
    10
>> whos                   %显示已输入的数据类型、字节大小
```

```

Name      Size      Bytes  Class
x         1x1         4  single array
y         1x1         8  double array
z         1x1         8  double array
Grand total is 3 elements using 20 bytes
>> isfloat(z)           %判断输入的数据是否为浮点型，是则返回“1”，否则返回“0”
ans =
     1
>> ones(3,'single')     %数据类型为单精度的全1矩阵生成
ans =
     1     1     1
     1     1     1
     1     1     1

```

realmin('single')和 realmin('double')函数，分别返回数据类型单精度浮点型和双精度浮点型的最小值，realmax('single')和 realmax('double')分别返回其相应的最大值。

【例 2.4】不同浮点型数据类型数值范围的计算。

```

>> realmin('single')    %单精度浮点型数值范围的下限
ans =
1.1754944e-038
>> realmin('double')    %双精度浮点型数值范围的下限
ans =
2.225073858507201e-308
>> realmax('single')    %单精度浮点型数值范围的上限
ans =
3.4028235e+038
>> realmax('double')    %双精度浮点型数值范围的上限
ans =
1.797693134862316e+308

```

2.3 逻辑类型

MATLAB 7.0 中的逻辑类型仅包括两个值：“0”和“1”，分别代表逻辑“假”和“真”。逻辑类型主要用于关系和逻辑运算，在使用过程中通过查找、条件语句的逻辑判断，可以判断条件是否为真。

【例 2.5】逻辑类型数据在编程中的一些应用实例。

(1) if 语句的条件判断。

```

a=5;
>> if a>0               %判断 a>0 是否为逻辑真
    disp('大于0')
else
    disp('小于0')
end
大于0

```

(2) 查找矩阵中符合一定条件的数据。

```

>> a=magic(3)
a =
     8     1     6
     3     5     7
     4     9     2
>> a>5               %查找矩阵中大于5的元素，如果是则返回“1”，不是则返回“0”
ans =
     1     0     1
     0     0     1
     0     1     0

```

(3) 逻辑运算。

```
>> x=[1 2 3 -2 0 -1];
>> ~x           %逻辑非运算, x 中元素是 0 则返回“1”, 否则返回“0”
ans =
     0     0     0     0     1     0
```

2.4 字符串

MATLAB 中提供了字符串类型用于处理文本等字符型数据。字符串可以理解为字符的数组, 而字符在 MATLAB 存储空间中又与相应的 ASCII 码对应。

2.4.1 字符串的生成

字符串的生成主要是通过直接赋值法、已有字符串的连接和其他数据类型的转换三种形式。其中其他数据类型的转换将在本章 2.7 节中详细叙述。

1. 一维字符串

在一对单引号的中间放置字符数据, 即构成字符串类型的数据。字符串类型的数据每个字符占 2 字节的存储空间。

【例 2.6】一维字符串的生成。

(1) 直接赋值法。

```
>>str1='ABCDEF';           %定义字符串 str1, 其中的字符数据为 ABCDEF
>> class(str1)              % str1 的数据类型为字符串
ans =
char
>> whos str1                %每个字符占 2 字节的存储空间
Name      Size      Bytes  Class
str1      1x6        12    char array
Grand total is 6 elements using 12 bytes
```

(2) 连接法, 可以通过连接函数 `strcat()` 或者连接符 “[]”。

```
>> str2=strcat(str1,'STR')   %strcat() 函数用于水平连接字符串变量 str1 和字符串“STR”
str2 =
ABCDEFSTR
>> str2=strcat('STR ',str1)  %注意使用 strcat() 函数连接时, 字符串末尾的空格会自动去除
str2 =
STRABCDEF
>> str3=[str1,str2]          %使用连接符 “[ ]” 连接字符串
str3 =
ABCDEFABCDEFSTR
>> str2=['STR ',str1]        %注意使用连接符 “[ ]” 连接时, 字符串末尾的空格不去除
str2 =
STR  ABCDEF
```

2. 二维字符串

二维字符串的生成方法与一维类似, 不过使用连接符 “[]” 连接时, 二维字符串要求每行的字符有相同的列数, 如果每行列数不同会报错, 此时可以通过填空补足。

【例 2.7】二维字符串的生成。

```
>> str1=['ABC';'ab']        %连接的字符串列数不同, 程序报错
??? Error using ==> vertcat
All rows in the bracketed expression must have the same
number of columns.
>> str1=['ABC';'ab ']       %通过字符串末尾填空, 保持字符串每行列数相同
```

```

str1 =
ABC
ab
>> str2=char('ABC','ab')    % char() 创建二维字符串，列数不同的字符行，末尾自动留空
str2 =
ABC
ab
>> whos str1 str2           %字符串 str1 和 str2 在内存中占相同的存储空间
Name      Size      Bytes  Class
str1      2x3        12   char array
str2      2x3        12   char array
Grand total is 12 elements using 24 bytes
>> str3=strvcat(str1,str2)  %垂直连接字符串 str1 和 str2
str3 =
ABC
ab
ABC
ab

```

2.4.2 字符串操作函数

字符串类型变量无论在任何编程环境中都是经常使用的，而 MATLAB 已为用户提供了丰富的函数直接处理字符串，包括字符串的判断、访问、查找、替换、比较、大小写转换、执行。下面详细介绍这些操作的实现。

1. 字符串的判断

- `ischar(s)`: 判断变量 `s` 的数据类型是否为字符串，返回结果为逻辑变量，如果是则为“1”，否则为“0”。
- `isletter(s)`: 判断字符串 `s` 中每个字符元素是否为字母，返回结果为逻辑型的向量，“1”代表字符串相应位置的元素为字母，“0”代表字符串相应位置的元素不为字母。
- `isspace(s)`: 判断字符串 `s` 中每个字符元素是否为空格，返回结果为逻辑型的向量，“1”代表字符串相应位置的元素为空格，“0”代表字符串相应位置的元素不为空格。

【例 2.8】字符串的判断。

```

>> str1=['A B123','CD 5 ']    %字符串生成
str1 =
A B123CD 5
>> ischar(str1)              %判断数据的数据类型是否为字符串
ans =
1
>> isletter(str1)            %判断字符串中每个字符元素是否为字母
ans =
1     0     1     0     0     0     1     1     0     0     0     0
>> isspace(str1)             %判断字符串中每个字符元素是否为空格
ans =
0     1     0     0     0     0     0     0     1     1     0     1

```

2. 字符串的访问

字符串中相应字符的访问是通过下标法来实现的，即根据字符元素在字符串中的位置来访问，其中位置的确定包括行列坐标和线性索引坐标两种方式。在 MATLAB 中字符数组是按列存储的。

【例 2.9】字符串的访问。

```

>> str1=char('abc','d f',' lee3 ');    %生成字符串
>> str1
str1 =
abc
d f

```

```
lee3
>> str1(1,:) %访问字符串中的第1行元素
ans =
abc
>> str1(1,2) %访问字符串中的第1行第2列元素
ans =
b
>> str1([1,2]) %按线性索引访问
ans =
ad
>> str1([1,2],[2,3]) %按行列坐标索引访问
ans =
bc
f
```

3. 字符串的查找和替换

- `k = strfind(str, s)`: 在字符串 `str` 中查找字符 `s`, 如存在则返回字符在字符串 `str` 中出现的下标, 没有返回的 `k` 为空矩阵。
- `k = findstr(s1, s2)`: 函数查找与被查找元素与其在函数中的顺序无关, 即函数 `findstr(s1, s2)` 与 `findstr(s2, s1)` 结果是一样的, 在长字符串中查找短字符串, 存在则返回字符在字符串中出现的下标, 没有返回的 `k` 为空矩阵。
- `str=strrep(s1,s2,s3)`: 在字符串 `s1` 中查找字符串 `s2` 并将其替换为字符串 `s3`。
- `i = strmatch(s1,s2)`: 在字符串 `s1` 中匹配查找与字符串 `s2` 起始一致的字符行, 返回行号。
- `i = strmatch(s1,s2,'exact')`: 在字符串 `s1` 中匹配查找与字符串 `s2` 完全一致的字符行, 返回行号。

【例 2.10】字符串的查找和替换。

```
>> str1='Specify character data by placing characters inside a pair of single quotes';
>> k = strfind(str1, 'ar') %在字符串 str1 中查找字符串“ar”，返回其在字符串中出现的位置
k =
    11     37
>> k = strfind(str1, 'are') %在字符串 str1 中查找字符串“are”，如果不存在，则返回空矩阵
k =
    []
>> k = findstr(str1, 'ar') %在字符串 str1 中查找字符串“ar”，返回其在字符串中出现的位置
k =
    11     37
>> k = findstr('ar',str1) %颠倒查找函数中的字符串顺序后，结果是一样的
k =
    11     37
>> str=strrep(str1,'ar','are')
str =
Specify chareacter data by placing chareacters inside a pair of single quotes
>> i = strmatch('min', strvcats('min', 'minimax', 'maximum')) %字符串中匹配查找
i =
     1
     2
>> i = strmatch('min',strvcats('min','minimax','maximum'),'exact') %字符串中精确的匹配查找
i =
     1
```

4. 字符串的比较

- `k=strncmp(s1,s2)`: 比较字符串 `s1` 和 `s2` 是否相同, 如果相同则返回逻辑变量“1”, 如果不相同则返回“0”。
- `k=strncmp(s1,s2,n)`: 比较字符串 `s1` 和 `s2` 前 `n` 个字符是否相同, 如果相同则返回逻辑变量“1”, 如果不相同则返回“0”。

- `k=strcmpi(s1,s2)`: 比较字符串 `s1` 和 `s2` 是否相同, 不区分字符串字母的大小写, 相同则返回逻辑变量 “1”, 不相同则返回 “0”。

【例 2.11】字符串的比较。

```
>> str1='ABC';
>> str2='ABC';
>> str3='abc';
>> str4='ABCD';
>> k1=strcmp(str1,str2)      %比较字符串 str1 和 str2 是否相同, 如果相同, 则返回 “1”
k1 =
    1
>> k2=strcmp(str1,str3)      %比较字符串 str1 和 str3 是否相同, 如果不相同, 则返回 “0”
k2 =
    0
>> k3=strcmpi(str1,str3)     %不区分大小写, 比较字符串 str1 和 str3 是否相同, 如果相同则返回 “1”
k3 =
    1
>> k4=strncmp(str1,str4,3)   %比较字符串 str1 和 str2 前 3 个字符是否相同, 如果相同, 则返回 “1”
k4 =
    1
```

5. 字符串的大小写转换

- `str=lower(s)`: 将字符串 `s` 中的大写英文字母全部转换为小写。
- `str=upper(s)`: 将字符串 `s` 中的小写英文字母全部转换为大写。

【例 2.12】字符串的大小写转换。

```
>> str1='asDD ssDD , kDEF';
>> str=lower(str1)          %将字符串 str1 中的大写英文字母全部转换为小写
str =
asdd ssdd , kdef
>> str=upper(str1)          %将字符串 str1 中的小写英文字母全部转换为大写
str =
ASDD SSDD , KDEF
```

6. 字符串的执行

`eval()`函数可用于字符串表达式的执行, 函数的具体用法如下。

- `eval(expression)`: 用于在命令行执行 `expression` 中的字符串表达式。
- `[a1,a2,a3,...] = eval('function(b1,b2,b3,...)')`: 其中 “`function(b1,b2,b3,...)`” 为待执行的字符串表达式, “`a1,a2,a3,...`” 为字符串表达式的输出结果。

【例 2.13】字符串的执行。

```
>> A=rand(3);
>> eval('length(A)')        %执行求取矩阵 A 长度的字符串表达式
ans =
    3
>> [I,J]=eval('size(A)')    %执行求取矩阵 A 大小的字符串表达式, 返回结果为矩阵 A 的行、列数
I =
    3
J =
    3
>> for i=1:3                %代码实现分别给变量 a1、a2、a3 赋值
    eval(['a',num2str(i),'=2*i'])
end
a1 =
    2
a2 =
    4
a3 =
    6
```

7. 字符串的空格操作相关函数

- `str=strtok(s)`: 查找字符串第一个空格前的字符, 返回到字符串 `str` 中。
- `str=deblank(s)`: 去除字符串 `s` 末尾的空格, 返回去除空格的字符串 `str`。
- `str = strtrim(s)`: 删除字符串 `s` 头尾的空格, 返回去除空格的字符串 `str`。
- `blanks(n)`: 生成含 `n` 个空格的字符串。

【例 2.14】字符串的空格操作。

```
>> str1='AABB CCDD EEFF ';
>> str2=strtok(str1)    %返回字符串 str1 第一个空格前的字符
str2 =
AABB
>> str3=deblank(str1)   %去除字符串末尾的空格, 原字符串长度为 16, 去除空格后为 14
str3 =
AABB CCDD EEFF
>> length(str1)
ans =
    16
>> length(str3)
ans =
    14
>> str4='  AABB CCDD EEFF ';
>> str5=strtrim(str4)   %去除字符串头、尾的空格, 原字符串长度为 18, 去除空格后为 14
>> str5
str5 =
AABB CCDD EEFF
>> length(str4)
ans =
    18
>> length(str5)
ans =
    14
>> str6=[str5,blanks(3),str5] %字符串中添加 3 个空格
str6 =
AABB CCDD EEFF   AABB CCDD EEFF
```

2.5 元胞数组

元胞数组是由可以包括任何数据类型的元胞组成的数组。通过元胞数组的使用, 可以在同一个变量中存储不同数据类型的数据, 给代码的编写带来很大便利。

2.5.1 元胞数组的创建

元胞数组的创建方法主要有直接赋值法和函数法。直接赋值法是指直接在命令行中给元胞数组的每个元素赋值, 或者使用大括号 “{}” 创建元胞数组。函数法是指使用 MATLAB 提供的 `cell()` 函数创建元胞数组。

1. 直接赋值法

元胞数组的直接赋值可以使用小括号 “()” 括起元胞的下标, 此时元胞中的内容需要使用大括号 “{}” 括起来。如果使用大括号 “{}” 括起元胞的下标时, 元胞中的内容无须另加标点, 与一般数组的元素输入相同。这两种方法的效果是一样的, 但要注意符号前后的配合。

使用大括号 “{}” 创建元胞数组的方法类似于使用中括号 “[]” 生成一般的数组, 行之间元素用分号 “;” 分隔, 列之间的元素用逗号 “,” 或者空格分隔。

【例 2.15】直接赋值法创建元胞数组。

```
%元胞下标使用小括号创建元胞数组
>> a{1,1}=8;
>> a{1,2}='cell study';
>> a{2,1}=magic(3);
>> a{2,2}=magic(3)>5;
>> a
a =
      [      8]      'cell study'
 [3x3 double] [3x3 logical]
%元胞下标使用大括号创建元胞数组
>> b(1,1)={8};
>> b(1,2)={'cell study'};
>> b(2,1)={magic(3)};
>> b(2,2)={magic(3)>5};
>> b
b =
      [      8]      'cell study'
 [3x3 double] [3x3 logical]
%使用大括号“{ }”创建元胞数组
>> {8, 'cell study';magic(3),magic(3)>5}
ans =
      [      8]      'cell study'
 [3x3 double] [3x3 logical]
```

2. 函数法

利用 cell() 函数生成元胞数组的过程，可以理解为一个先利用函数对元胞内存空间预分配，然后对元胞中的元素进行赋值。Cell() 函数的用法如下。

- $x = \text{cell}(n)$: 生成 $n \times n$ 的元胞数组 x 。
- $x = \text{cell}(m,n)$ 或者 $x = \text{cell}([m \ n])$: 生成 $m \times n$ 的元胞数组 x 。
- $x = \text{cell}(m,n,p,...)$ 或者 $x = \text{cell}([m \ n \ p \ ...])$: 生成 $m \times n \times p$ 的元胞数组 x 。
- $x = \text{cell}(\text{size}(A))$: 生成与数据 A 具有相同大小的元胞数组 x 。

【例 2.16】函数法创建元胞数组。

```
>> a=cell(2) %生成 2×2 的元胞数组 a
a =
      []      []
      []      []
>> a={1,'c';2,cell(3)};%给生成的元胞数组赋值
>> a
a =
      [1]      'c'
      [2]      {3x3 cell}
```

2.5.2 元胞数组的访问

MATLAB 7.0 中对元胞数组的访问提供了大括号和小括号两种方式，其中大括号访问的是元胞数组中元胞的内容，可以对元胞中的内容进行进一步的操作，而小括号访问的是元胞数组的元胞，是个整体，无法对元胞中的具体数据进行操作。

1. 利用大括号访问元胞数组

利用大括号可以访问到元胞数组内元素具体的内容，并可对其中的数据执行具体的操作。

【例 2.17】利用大括号访问元胞数组。

```
>> a={8, 'cell study';magic(3),magic(3)>5} %创建元胞数组
a =
      [      8]      'cell study'
```

```
[3x3 double]    [3x3 logical]
>> a{1,1}          %访问元胞数组的第1行第1列的元胞内容
ans =
    8
>> a{2,1}          %访问元胞数组的第2行第1列的元胞内容
ans =
    8     1     6
    3     5     7
    4     9     2
>> a{1,:}          %访问元胞数组的第1行中的所有元胞内容
ans =
    8
ans =
cell study
>> a{:,2}          %访问元胞数组的第2列中的所有元胞内容
ans =
cell study
ans =
    1     0     1
    0     0     1
    0     1     0
>> a{2,1}(2,2)     %访问元胞数组内元胞子数据的内容
ans =
    5
>> a{1,1}+8
ans =
   16
```

2. 利用小括号访问元胞数组

利用小括号访问元胞数组是访问元胞数组中元胞的整体，不可以访问元胞数组中的具体元素运算。

【例 2.18】利用小括号访问元胞数组。

```
>> a={8, 'cell study';magic(3),magic(3)>5};%创建元胞数组
>> a(1,1)          %访问到元胞数组的元胞整体
ans =
    [8]
>> a(1,1)+8        %不可以对元胞数组的元胞整体进行运算
??? Function 'plus' is not defined for values of class 'cell'.
Error in ==> plus at 14
    builtin('plus', varargin{:});
```

2.5.3 元胞数组的显示

MATLAB 中提供了 `celldisp()` 和 `cellplot()` 函数用于显示元胞数组。`Celldisp()` 函数可以显示元胞数组的具体内容，而 `cellplot()` 函数以图形方式显示元胞数组。函数的具体用法如下。

- `celldisp(s)`: 用于显示元胞数组 `s` 中的具体内容。
- `celldisp(s,name)`: 以字符串 `name` 为元胞名，显示元胞数组 `s` 中的具体内容。
- `cellplot(s)`: 以图形化的方式显示元胞数组 `s`。
- `cellplot(s,'legend')`: 以图形化的方式显示元胞数组 `s`，同时显示不同数据类型的颜色图例标注。

【例 2.19】元胞数组的显示。

```
>> a={8, 'cell study';magic(3),magic(3)>5};
>> celldisp(a)      %显示元胞数组 a 中的具体内容
a{1,1} =
    8
a{2,1} =
```

```

      8      1      6
      3      5      7
      4      9      2
a{1,2} =
cell study
a{2,2} =
      1      0      1
      0      0      1
      0      1      0
>> celldisp(a,'s')           %以 s 为元胞数组名，显示元胞数组 a 中的具体内容
s{1,1} =
      8
s{2,1} =
      8      1      6
      3      5      7
      4      9      2
s{1,2} =
cell study
s{2,2} =
      1      0      1
      0      0      1
      0      1      0
>> cellplot(a)              %以图形化的方式显示元胞数组的内容，如图 2.1 所示。

```

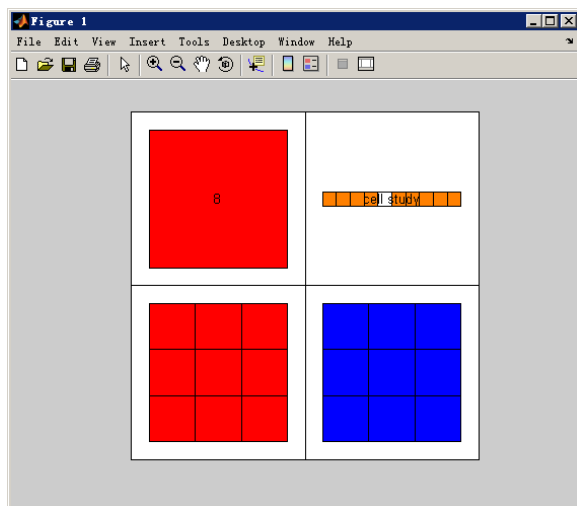


图 2.1 以图形化的方式显示元胞数组

2.5.4 元胞数组的删除

元胞数组的删除主要是通过把需要删除的元胞赋值为空来实现。通过大括号和小括号访问元胞并将其置空，可以分别删除元胞数组的内容或整体。

1. 利用大括号删除元胞数组的内容

利用大括号可以删除元胞数组元胞内的具体内容，可以删除单个元胞及其中的具体元素，但不可删除一行或一列元胞数组内容。

【例 2.20】利用大括号删除元胞数组的内容。

```

>> a={8, 'cell study';magic(3),magic(3)>5};
>> a{1,1}=[];                %删除元胞数组内单个元胞
>> a
a =
[]      'cell study'

```

```
[3x3 double]    [3x3 logical]
>> a{2,1}=[];    %删除元胞数组内单个元胞
>> a
a =
    []    'cell study'
    []    [3x3 logical]
>> a{2,2}(1,:)=[];    %删除元胞数组内单个元胞的具体元素
>> a
a =
    []    'cell study'
    []    [2x3 logical]
>> a{1,:}=[]    %不可以通过大括号删除元胞数组的一行或一列元素
??? Insufficient outputs from right hand side to satisfy comma separated
list expansion on left hand side. Missing [] are the most likely cause.
```

2. 利用小括号删除元胞数组的元胞

利用小括号可以删除元胞数组的元胞整体，可以删除一行或一列的元胞，但不可以删除其中具体的一个元胞。

【例 2.21】利用小括号删除元胞数组的内容。

```
>> a={8, 'cell study';magic(3),magic(3)>5};
>> a(1,1)=[];    %删除单个元胞整体，出错
??? A null assignment can have only one non-colon index.
>> a(1,:)=[];    %删除元胞数组的一行
>> a
a =
    [3x3 double]    [3x3 logical]
>> a(:,2)=[]    %删除元胞数组的一列
a =
    [3x3 double]
```

2.6 结构体

结构体数据类型可以把不同数据类型的变量放到同一个变量名下，通过不同“域”的概念对结构体中的不同数据进行赋值、操作。结构体中的数据存储在相应的“域”中。

2.6.1 结构体的生成

结构体类型数据生成的方法主要有命令行直接赋值法和函数法。命令行直接赋值法是指直接把不同类型的数据赋值给结构体变量不同的域中；函数法是指通过 MATLAB 中自带的函数 `struct()` 创建结构体变量。下面分别通过这两种方法来创建结构体类型的变量。

1. 命令行直接赋值法

通过命令行直接赋值即直接给结构体中的不同域赋值，结构与域之间用点号“.”连接，不同域中可保存不同类型的数据。下面以实例演示通过命令行直接创建结构体变量。

【例 2.22】命令行直接赋值法生成结构体。

```
>> ss.str='ABC';
>> ss.num=[5:2:16];
>> ss.str2=['A','S','BS'];
>> ss
ss =
    str: 'ABC'
    num: [5 7 9 11 13 15]
    str2: 'ASBS'
```

2. 函数法

MATLAB 中提供了 struct() 函数创建结构体，具体用法如下。

```
s = struct('field1', values1, 'field2', values2, ...)
```

其中，“field1”和“field2”为域名，“values1”和“values2”为域中的值。

【例 2.23】函数法生成结构体。

```
>> ss=struct('str','ABC','num',[5:2:16],'str2',['A','S','BS'])
ss =
    str: 'ABC'
    num: [5 7 9 11 13 15]
   str2: 'ASBS'
```

2.6.2 结构体的操作

结构体的常用操作包括对结构体元素的访问、结构体的显示、结构体的删除。下面将具体叙述结构体的常用操作。

1. 访问结构体中的元素

【例 2.24】访问结构体中的元素。

```
>> ss=struct('stu_name',{'wang','ma','li'},'stu_num',{100,101,102},'stu_age',{22,24,22})
ss =
1x3 struct array with fields:
    stu_name
    stu_num
    stu_age
>> ss(:,1)           %访问结构体 ss 中的第一列元素
ans =
    stu_name: 'wang'
    stu_num: 100
    stu_age: 22
>> ss.stu_name %访问结构体 ss 中的域 stu_name
ans =
wang
ans =
ma
ans =
li
>> ss(1).stu_name %访问结构体 ss 中第 1 列的域 stu_name
ans =
wang
```

2. 结构体的显示

MATLAB 中提供了 fieldnames() 函数显示结构体的域名，而 getfield() 函数可用于显示结构体各域中的具体内容。这两个函数的具体用法如下。

- names = fieldnames(s): 返回结构体 s 中的各域名到变量 names。
- f = getfield(s,'field'): 显示结构体 s 中域 field 的具体内容，返回到变量 f 中，s 必须为 1×1 的结构体。

【例 2.25】结构体的显示。

```
>> ss=struct('stu_name',{'wang','ma','li'},'stu_num',{100,101,102},'stu_age',{22,24,22});
>> names = fieldnames(ss)           % fieldnames() 函数显示结构体 ss 中的域名
names =
    'stu_name'
    'stu_num'
    'stu_age'
```

```
>> ss2=struct('str','ABC','num',[5:2:16],'str2',['A','S','BS']);
>> whos ss ss2           %结构体 ss、ss2 的大小
Name      Size          Bytes Class

ss         1x3          796  struct array
ss2        1x1          434  struct array
Grand total is 39 elements using 1230 bytes
>> f = getfield(ss,'stu_name') %结构体 ss 大小 1×3 不可用函数 getfield 显示域中的内容
??? Illegal right hand side in assignment. Too many elements.
Error in ==> getfield at 38
    f = s.(deblank(strField)); % deblank field name
>> f = getfield(ss2,'str')    % ss2 大小 1×1 可用函数 getfield 显示域中的内容
f =
ABC
```

3. 结构体的删除

结构体的删除使用函数 `rmfield`，其具体用法如下。

- `s = rmfield(ss,'field')`: 用于删除结构体中的域 “field”。
- `s = rmfield(ss,FIELDS)`: 用于同时删除结构体中的多个域，`FIELDS` 为需要删除的多个域的域名的字符串。

【例 2.26】结构体的删除。

```
>> ss=struct('stu_name',{'wang','ma','li'},'stu_num',{100,101,102},
'stu_age',{22,24,22});
>> s = rmfield(ss,'stu_name')
s =
1x3 struct array with fields:
    stu_num
    stu_age
>> s2 = rmfield(ss',{'stu_name','stu_age'})
s2 =
1x3 struct array with fields:
    stu_num
```

2.7 不同数据类型之间的转化

MATLAB 中为用户提供了丰富的数据类型，用户可以根据实际问题的需要创建不同数据类型的变量，同时 MATLAB 也提供了各种函数用于不同类型数据之间的转换。下面将详细介绍不同类型数据之间常有的转化。

1. 数值类型的互相转换

数据类型的转换与不同数值类型变量定义的格式相同，如 `x2=class(x1)`，其中 `x1` 为原数据类型的数据，`x2` 为转换后数据类型的数据，`class` 为转换的数据类型。可以转换的数据类型包括所有的数值类型，有双精度、单精度浮点型，16 位、32 位、64 位无符号整型数据，16 位、32 位、64 位有符号整型数据等。

【例 2.27】数值类型的互相转换。

```
>> x=32;           %定义数值类型数据 x，默认状态下为双精度浮点型
>> x1=single(x);   %转换双精度浮点型数据为单精度浮点型数据
>> x2=uint8(x);    %转换双精度浮点型数据为无符号 8 位整型数据
>> x3=int8(x);     %转换双精度浮点型数据为有符号 8 位整型数据
>> whos           %查看各变量的数据类型
Name      Size          Bytes Class
x         1x1           8  double array
```



```

x1          1x1          4 single array
x2          1x1          1 uint8 array
x3          1x1          1 int8 array
Grand total is 4 elements using 14 bytes

```

2. 字符串与数值类型的互相转换

字符串与数值类型的互相转换包括数组与字符串的转换，1~127 的 ASCII 码值转换为字符，还有不同进制数据之间的转换等。

数组与字符串的转换通过 num2str()和 str2num()函数实现，具体的函数用法如下。

- `str = num2str(A)`: 把数值型数据数组 A 转换为字符型数据，默认情况下转换的数据精度为 5 位有效数字。
- `str = num2str(A,precision)`: 按照指定的数据精度转换数组 A 到字符串类型数据 str，precision 为字符串中数据的有效位数。
- `str = num2str(A,format)`: 按照指定的数据格式转换数组 A 到字符串类型数据 str。format 的书写格式为 %m1.m2g/f/e，其中 m1 指定总共显示的有效数字位数，m2 代表小数点后的有效数字位数，“g”格式代表用指数或定点标记，“e”格式代表用指数标记，“f”格式代表用定点标记，与函数 sprintf()的输出显示设置相同。
- `x = str2num('str')`: 字符串转化为数组。

【例 2.28】字符串与数值类型的互相转换

```

>> a=rand(3);           %生成随机矩阵 a
>> s=num2str(a)         %数组 a 转换为字符串 s
s =
0.95013      0.48598      0.45647
0.23114      0.8913      0.018504
0.60684      0.7621      0.82141
>> s=num2str(a,3)       %数组 a 转换为字符串 s，保留小数点后 3 位
s =
0.95      0.486      0.456
0.231      0.891      0.0185
0.607      0.762      0.821
>> s=num2str(a,'%3.2e') %转换为字符串 s，保留 3 位有效数字，小数点后留 2 位，以指数形式输出
s =
9.50e-0014.86e-0014.56e-001
2.31e-0018.91e-0011.85e-002
6.07e-0017.62e-0018.21e-001
>> s=num2str(a,'%3.2g') %转换为字符串 s，保留 3 位有效数字，小数点后留 2 位，以“g”格式输出
s =
0.950.490.46
0.230.890.019
0.610.760.82
>> s=num2str(a,'%3.2f') %转换为字符串 s，保留 3 位有效数字，小数点后留 2 位，以“f”格式输出
s =
0.950.490.46
0.230.890.02
0.610.760.82

```

另外，int2str()函数和 str2int()函数可以完成整型数据与字符串的转换，即取整数据与字符串的转换。mat2str()函数和 str2mat()函数可以实现矩阵与字符串的转换，其用法类似于 num2str()函数和 str2num()函数，但不可以用于高维数组，可以参见 MATLAB 的帮助文档。

ASCII 码值与字符之间的转换通过 char()函数和 abs()函数来实现。

【例 2.29】ASCII 码值与字符之间的转换。

```

>> abs('A')           %字符转换为 ASCII 码值
ans =
65

```

```
>> char(65)           % ASCII 码值转换为字符
ans =
A
>> char([65:127])    %多个 ASCII 码值转换为字符
ans =
ABCDEFGHIJKLMNOPQRSTUVWXYZ[\]^_`abcdefghijklmnopqrstuvwxyz{|}~
```

MATLAB 中还提供了一些函数用于不同进制数据之间的转换, 其中涉及十进制数与其他各进制数字字符串之间的转换, 主要涉及的函数如下。

- dec2hex(x)和 hex2dec(x): 用于十六进制数和十进制数之间的相互转换。
- dec2bin(x)和 bin2dec(x): 用于二进制数和十进制数之间的相互转换。
- str = dec2base(d,base)和 d = base2dec('strn',base): 用于任意进制的数与十进制数之间的转换。

【例 2.30】不同进制数据之间的转换。

```
>> s=dec2hex(59)       %十进制数 59 转换为十六进制数
s =
3B
>> n=hex2dec('1B')    %十六进制数“1B”转换为十进制数
n =
27
>> s=dec2bin(19)       %十进制数 19 转换为二进制数
s =
10011
>> n=bin2dec('100011') %二进制数“100011”转换为十进制数
n =
35
>> s=dec2base(33,8)    %十进制数 33 转换为八进制
s =
41
>> n=base2dec('222',8) %十进制数“222”转换为十进制
n =
146
```

3. 元胞数组与数值类型的互相转换

元胞数组与数值类型的互相转换通过 num2cell()函数来实现, 具体用法如下。

- c = num2cell(A): 转换数组 A 到元胞数组 c。
- c = num2cell(A,dims): 按照指定的维数转换数组 A 到元胞数组 c。

【例 2.31】元胞数组与数值类型的互相转换。

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> num2cell(A)         %转换数组 A 到元胞数组
ans =
     [8]     [1]     [6]
     [3]     [5]     [7]
     [4]     [9]     [2]
>> num2cell(A,[1,9])   %按指定维数转换数组 A 到元胞数组
ans =
 [3x1 double] [3x1 double] [3x1 double]
```

4. 字符串、元胞数组、结构体之间的互相转换

函数 char 和 cellstr 用于元胞数组和字符串之间的转换; 函数 cell2struct 和 struct2cell 可用于元胞数组和结构体的转换。下面以实例来具体演示这些函数的使用。

【例 2.32】字符串、元胞数组、结构体之间的互相转换。

```

>> str2CELL=cellstr({'abc', 'def';'ABC','DEF'})    %字符串转换为元胞数组
str2CELL =
    'abc'    'def'
    'ABC'    'DEF'
>> class(str2CELL)
ans =
cell
>> cell2str=char(Str2cell)                        %元胞数组转换为字符串
cell2str =
abc
ABC
def
DEF
>> class(cell2str)
ans =
char
>> c2s=cell2struct(str2CELL,'A',4)                %元胞数组转换为结构体
c2s =
2x2 struct array with fields:
    A
>> class(c2s)
ans =
struct
>> s2c=struct2cell(c2s)                            %结构体转换为元胞数组
s2c(:, :, 1) =
    'abc'    'ABC'
s2c(:, :, 2) =
    'def'    'DEF'
>> class(s2c)
ans =
cell

```

2.8 本章小结

本章主要向用户介绍了 MAXTLAB 常用数据类型整型、浮点型、逻辑类型、字符串、元胞数组、结构体类型的基本知识，及其不同数据类型之间的转化。

通过本章的学习，相信读者已对 MATLAB 常用数据类型有了比较好的了解，今后在实际应用中选择什么数据类型还要具体问题具体分析，采用最合理的数据类型将有利于算法的分析执行，提高工作效率，同时数据类型之间的相互转换也更有利于我们使用不同的数据类型。



第3章

矩阵和数组

MATLAB 是基于矩阵和数组计算的，可以直接对矩阵和数组进行整体的操作，而一般的高级程序设计语言没有矩阵的概念，对于数组的操作也只能通过循环每次对数组内的一个数据操作，这是 MATLAB 与其他程序设计语言最大的区别，也是最大的优势之一。矩阵语言操作的使用能有效提高编程效率，同时矩阵为数学上的概念，与一般算法定义中的数据概念很接近，使用矩阵作为基本编程数据单位，可以较为简单地实现复杂算法，可以直接进行矩阵的运算。本章主要讲述矩阵、数组的基本操作，涉及矩阵和数据创建、简单运算、特殊运算、向量和高维数组的基本知识。通过本章的学习将使读者掌握 MATLAB 最基本的数据结构的操作。

3.1 矩阵和数组的概念

数组为具有相同数据类型的数据组合，矩阵的概念主要应用于数学中，在 MATLAB 中矩阵一般即指二维数组，但是矩阵与数组在部分运算上又是有很大区别的，因此对于矩阵与数组共性的操作（比如矩阵和数组的创建等）将放在一起介绍，此部分对于矩阵的操作同样适用于数组；而矩阵与数组有区别的运算部分，将独立出来详细描述。

3.2 矩阵和数组的创建

矩阵是 MATLAB 软件学习操作的核心，本节将从最基本的矩阵和数组的创建开始，介绍矩阵和数组的使用。同时矩阵与二维数组在生成上的操作是相同的，因此本节中涉及矩阵的操作也同样适合于数组。

矩阵和数组的生成方法常用的主要有直接输入、函数、外部导入法这 3 种。下面详细介绍这 3 种方法的使用。

1. 直接输入法

直接输入法即在命令行或 m 文件中直接定义变量及其中的数据内容，适用于小数据量矩阵和数组的创建。关于直接输入法创建矩阵和数组主要有以下几条规则：

- 同一行中的数据使用空格或者逗号分隔。
- 分号表示每一行数据输入结束。
- 所有数据都包含在方括号 “[]” 中。

【例 3.1】 矩阵的输入。

在命令窗口中输入：

```
A=[1 2 3 4; 5 6 7 8; 9 10 11 12]
```

按 Enter 键后, 命令窗口中将会出现显示输入的数据矩阵, 同时在 MATLAB 工作空间内将存有矩阵 A。如果矩阵过大, 希望节省数据输入时间, 或者因某些原因不希望输入的矩阵在命令窗口中显示出来, 则在输入矩阵后加上 “;”。

```
A =
     1     2     3     4
     5     6     7     8
     9    10    11    12
```

2. 函数法

函数法主要用于一些具有特殊规律的矩阵, 例如全零、全 1、上三角、下三角矩阵等。MATLAB 提供了生成这些特殊矩阵的函数, 直接调用函数即可生成相应的矩阵, 下面具体介绍常用的特殊矩阵函数法生成的过程。

(1) zeros()函数: 全零矩阵生成。MATLAB 语言没有数据声明的要求, 自动扩展变量的存储空间, 对于大型矩阵操作, 在计算过程中重复地扩大内存空间, 会影响计算的速度。全零矩阵一般用于初始变量创建时预留内存空间, 在后面的计算中再为矩阵元素赋具体的值。其中,

- zeros(n): 用于生成 $n \times n$ 的全零矩阵。
- zeros(m,n): 用于生成 $m \times n$ 的全零矩阵。
- zeros(size(a)): 用于生成和矩阵 a 具有同样大小的数据矩阵。
- zeros(d1,d2,d3...): 用于生成 $d1 \times d2 \times d3$ 的全零矩阵。
- zeros(m, n, classname): 用于生成指定数据类型的 $m \times n$ 的全零矩阵。

【例 3.2】全零矩阵的生成。

```
>> zeros(3)                                %生成 3×3 的全零矩阵
ans =
     0     0     0
     0     0     0
     0     0     0

>> zeros(3,4)                              %生成 3×4 的全零矩阵
ans =
     0     0     0     0
     0     0     0     0
     0     0     0     0

>> x = zeros(2,3,'int8')                    %生成 2×3 的整型全零矩阵
x =
     0     0     0
     0     0     0
```

(2) eye()函数: 单位矩阵生成。eye()函数的用法与全零矩阵生成方法类似, 但是 eye()函数不支持二维以上矩阵的生成。

【例 3.3】单位矩阵的生成。

```
>> y=eye(2,3)                              %生成 2×3 的单位矩阵
y =
     1     0     0
     0     1     0

>> y=eye(3)                                %生成 3×3 的单位矩阵
y =
     1     0     0
     0     1     0
     0     0     1
```

(3) ones: 全 1 矩阵生成, 用法同 zeros()函数。

【例 3.4】全 1 矩阵的生成。

```
>> ones(3)                                %生成 3×3 的全 1 矩阵
ans =
     1     1     1
```

```

1     1     1
1     1     1

```

(4) rand()函数: 随机矩阵生成。Rand()函数只用于生成 0~1 的平均分布的随机数, 不包括 0 和 1, 如果希望生成在其他范围内的随机数, 则需要编写一定的代码, 下面的例子中会演示。同时 rand()函数随机数的生成并不是完全的随机, 依赖于随机数生成的种子数, 在实际应用中可以固定种子数, 使每次生成具有随机分布的相同随机数。

- rand(n): 用于生成 $n \times n$ 的随机矩阵。
- rand(m,n): 用于生成 $m \times n$ 的随机矩阵。
- rand(m,n,p,...): 用于生成 $m \times n \times p \dots$ 的高维随机矩阵。
- rand(size(A)): 用于生成和矩阵 A 有相同大小的随机矩阵。
- rand('state',s): 设定生成随机数的种子数 s。

【例 3.5】随机数的生成。

%设定种子数, 产生特定种子数下相同的随机数

```
>> rand('state',0);
```

```
>> rand(3)
```

```
ans =
```

```

0.9501    0.4860    0.4565
0.2311    0.8913    0.0185
0.6068    0.7621    0.8214

```

%产生区间 (1,100) 内的随机数

```
>> a = 1; b = 100;
```

```
x = a + (b-a) * rand(3)
```

```
x =
```

```

45.0256    92.2595    41.1649
61.9278    74.0825    93.6115
79.4018    18.4503    91.7735

```

%产生 50 个区间 [1,100] 内的随机正整数

```
>> a = 1; b = 100;
```

```
x = a + fix(b* rand(1,50))
```

```
x =
```

```
Columns 1 through 16
```

```
35    54    73    31    84    57    38    71    55    45    70    63    80    96    53    89
```

```
Columns 17 through 32
```

```
18    98    28    26    88    74    14     2    90    20    30    67    29    47     7    99
```

```
Columns 33 through 48
```

```
59    43    52    34    44    23    58    77    53    65    21    38    79    69    47    57
```

```
Columns 49 through 50
```

```
80     6
```

相近函数扩展。

- randn: 用于生成均值为 0, 方差为 1 的正态分布的随机数, 其他用法同 rand()函数。
 - randperm(n): 用于生成 1:n 随机分布的 n 个正整数。
- (5) compan()函数: 生成伴随矩阵, 只适用于向量。
- compan(u): u 是多项式的系数向量, 高次项系数在前, 低次项系数在后, 按顺序排列。

【例 3.6】多项式 x^3-7x+6 的伴随矩阵生成。

```
>> u = [1 0 -7 6];
```

```
A = compan(u) %伴随矩阵的生成
```

```
A =
```

```

0     7    -6
1     0     0
0     1     0

```

%伴随矩阵可用于进一步求方程 $x^3-7x+6=0$ 的根

```
>> eig(A)
```

```
ans =
```

```
-3.0000
```

```
2.0000
1.0000
```

(6) `magic()`函数: 生成魔方矩阵, 矩阵每行、每列及两条对角线上元素和都相等。

- `magic(n)`: 生成 $n \times n$ 魔方矩阵。

【例 3.7】 魔方矩阵的生成。

```
>> magic(3)           %生成 3×3 的魔方矩阵
ans =
     8     1     6
     3     5     7
     4     9     2
```

(7) `diag()`函数: 生成对角矩阵, 即只有对角线上有非零元素的矩阵。

- `diag(X)`: X 为一向量, m 为向量长度, 用于生成一个 $m \times m$ 对角矩阵, 主对角线元素即为向量 X 的元素。
- `diag(X,k)`: X 为一向量, 用于生成一个 k 对应的对角线上为向量 X 的对角矩阵, 其中主对角线 $k=0$, 位置向上或向下对角线 k 相应加 1 或减 1。

【例 3.8】 对角矩阵的生成。

```
>> diag(1:5)           %对角线上为向量 1:5 的对角矩阵生成
ans =
     1     0     0     0     0
     0     2     0     0     0
     0     0     3     0     0
     0     0     0     4     0
     0     0     0     0     5

>> diag(1:5,1)         %k 为 1 对应的对角线上为向量 1:5 的对角矩阵生成
ans =
     0     1     0     0     0     0
     0     0     2     0     0     0
     0     0     0     3     0     0
     0     0     0     0     4     0
     0     0     0     0     0     5
     0     0     0     0     0     0

>> diag(1:5,-1)        %k 为 -1 对应的对角线上为向量 1:5 的对角矩阵生成
ans =
     0     0     0     0     0     0
     1     0     0     0     0     0
     0     2     0     0     0     0
     0     0     3     0     0     0
     0     0     0     4     0     0
     0     0     0     0     5     0
```

(8) `triu`: 上三角矩阵, 即对角线以下元素全为 0 的矩阵。

- `triu(X)`: 用于生成矩阵 X 的上三角矩阵。
- `triu(X,k)`: 用于生成矩阵 X 的上三角矩阵, 其中 k 对应的对角线以下的元素全为 0。

(9) `tril`: 下三角矩阵, 即对角线以上元素全为 0 的矩阵。

- `tril(X)`: 用于生成矩阵 X 的下三角矩阵。
- `tril(X,k)`: 用于生成矩阵 X 的下三角矩阵, 其中 k 对应的对角线以上的元素全为 0。

【例 3.9】 上三角、下三角矩阵的生成。

```
%上三角矩阵的生成
>> triu(ones(3,3))
ans =
     1     1     1
     0     1     1
     0     0     1

>> triu(ones(3,3),1)
ans =
```

```
0     1     1
0     0     1
0     0     0
%下三角矩阵的生成
>> tril(ones(3,3))
ans =
     1     0     0
     1     1     0
     1     1     1
>> tril(ones(3,3),-1)
ans =
     0     0     0
     1     0     0
     1     1     0
```

以上仅介绍了 MATLAB 语言中一些常见函数生成的使用,对于其他函数,例如 pascal(Pascal 矩阵生成)、hadamard(哈达玛矩阵生成)、hilb(希尔伯特矩阵生成)、invhilb(希尔伯特矩阵的逆矩阵生成)等,如有需要可以查阅相关帮助文件。

3. 外部导入法

外部导入法主要是指通过数据导入平台或者文件输入函数,把 txt、excel、mat 等文件中存储的数据导入 MATLAB 工作空间内,并以矩阵的形式存储数据。通过数据导入平台导入数据在本书第 1 章的 1.3.2 节中有详细的介绍;外部文件中,数据通过函数命令的导入参见第 10 章数据输入/输出部分。通过外部文件导入数据,主要用于在硬盘空间保存地点数据的导入,多是需要重复调用或者实际观察数据,或者直接生成比较麻烦的数据。

3.3 矩阵及数组的基本操作

在创建了基本的矩阵及数组后,需要对矩阵及数组进行简单的操作,包括矩阵编辑、数据读取、矩阵大小的求取、矩阵的拼接、形状变化、数据查找等,这部分操作虽然简单,但灵活掌握矩阵及数组的这部分基本操作,是熟练掌握 MATLAB 语言的重点。

3.3.1 基本信息获取

矩阵及数组基本信息的获取主要包括以下几种。

1. 数据的显示

disp(x)函数用于在命令窗口中输出矩阵 x,不显示矩阵名;而通过在命令行语句后不加分号也可以完成命令行数据的显示,具体的区别见例 3.10。

【例 3.10】命令窗口矩阵的显示。

```
x=rand(2,3)
%在语句中不加分号的显示
x =
    0.6072    0.3705    0.4514
    0.6299    0.5751    0.0439
%使用函数 disp 的显示
>> disp(x)
    0.6072    0.3705    0.4514
    0.6299    0.5751    0.0439
```

2. 矩阵的判断

isempty(A): 判断矩阵是否为空,若为空返回 1,否则返回 0;空矩阵是指没有任何元素的矩

阵，一般的使用过程中，对空矩阵执行操作会出错，因而在对矩阵进行一些操作前需要判断矩阵是否为空。

- `isequal(A,B)`: 判断矩阵 A、B 的数值是否相等，仅当 A、B 矩阵所有元素都相等时返回 1，否则返回 0。
- `isfloat(A)`: 判断矩阵 A 的数据类型是否为浮点型，如果是则返回 1，否则返回 0。
- `isinteger(A)`: 判断矩阵 A 的数据类型是否为整数型，如果是则返回 1，否则返回 0。

【例 3.11】矩阵的判断。

(1) 矩阵是否为空的判断。

```
>> B = rand(2,2);           %生成随机矩阵 B
>> isempty(B) %判断矩阵是否为空，返回 0，表明矩阵不为空
ans =
    0
>> B(:)=[];                %B 中元素删除
>> isempty(B) %B 中元素删除后，矩阵为空，返回 1
ans =
    1
```

(2) 矩阵是否相等的判断。

```
>> A=[1 0;0 1];
>> B=[1 0;0 1];
>> C=[1 1;0 0];
>> isequal(A,B)           %A、B 矩阵相同返回 1
ans =
    1
>> isequal(A,C)           %A、C 矩阵相同返回 0
ans =
    0
```

(3) 矩阵数据类型的判断。

```
>> A=[1 0;0 1]; %默认状况下 MATLAB 输入数据为浮点型，返回 1
>> isfloat(A)
ans =
    1
>> isinteger(A) %矩阵 A 数据类型为浮点型，不为整型，返回 0
ans =
    0
```

3. 大小信息的获取

- `size(X)`: 用于获取矩阵 X 的行数和列数。
- `length(X)`: 用于获取矩阵 X 的长度，即矩阵行数或列数中的较大值。
- `numel(X)`: 用于获取矩阵 X 中元素个数总和。
- `ndims(X)`: 用于获取矩阵 X 的维数。

```
>> x = rand(2,10);          %生成 2×10 的随机矩阵
>> [m,n]=size(x)           %获取矩阵各维的大小，其中 m 代表行数，n 代表列数
m =
    2
n =
   10
>> size(x,2)                %获取矩阵指定维数下的长度
ans =
   10
>> n = length(x)            %获取矩阵 x 的长度，即行列中的较大值 10
n =
   10
>> numel(x)                 %获取矩阵中所有元素个数的总和
ans =
   20
```

```
>> ndims(x)           %矩阵维数的确定
ans =
     2
```

3.3.2 元素访问

数据的标识指从矩阵中访问指定的数据，可以是单个元素、一行、一列、多个元素或者所有元素，在访问到矩阵指定的数据后将对其进行进一步的操作。

1. 单个元素的访问

矩阵和数组是由行和列组成的，例如矩阵 A 中 $A(i,j)$ 对应于矩阵的第 i 行、第 j 列的元素，可以通过数据行列形式访问指定行列下标的元素。同时 MATLAB 提供矩阵单下标的数据访问方式，这主要是因为基于矩阵的列存储数据的，第 i 行、 j 列的数据对应单下标即为 $(j-1)*m+i$ ，其中 m 为矩阵的行数。在访问到需要的单个元素，即可进一步对访问到的单个元素进行编辑操作。

【例 3.12】 单个元素的访问。

```
%创建 3×3 的魔术矩阵
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
%读取第 1 行、第 3 列的元素
A(1,3)
ans =
     6
%读取线性编码为 7 的元素，即为对应的第 1 行、第 3 列的元素
>> A(7)
ans =
     6
%访问第 1 行、第 3 列的元素，并对其赋新值
>> A(1,3)=0
A =
     8     1     0
     3     5     7
     4     9     2
>> A(2,4) %不可访问超过矩阵大小以外的元素
??? Index exceeds matrix dimensions.
```

2. 多个元素的访问

矩阵中多个元素的访问可以通过矩阵的下标或者根据矩阵中元素的线性索引值来访问，下面通过具体的实例来说明。

【例 3.13】 多个元素的访问。

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> B=A([1,2],[2,3])           %访问矩阵 A 中第 1 行和第 2 行中第 2、3 列的元素
B =
     1     6
     5     7
>> B=A([1,7,8])               %按单下标机制访问矩阵中的第 1、7、8 个元素
B =
     8     6     7
```

3. 行列元素的访问

行列元素的访问通过冒号运算符来实现，对于行数据访问，“:”代表所有列的数据，对于列数据访问，“:”代表所有行数据，即 $A(i,:)$ 访问矩阵的第 i 行的所有数据。 $A(:,j)$ 访问矩阵的第 j 列的所有数据。下面以实例演示具体的操作。

【例 3.14】行列元素的访问。

```
>> A=rand(3,4)           %创建 3×4 的随机矩阵 A
A =
    0.9218    0.4057    0.4103    0.3529
    0.7382    0.9355    0.8936    0.8132
    0.1763    0.9169    0.0579    0.0099
>> A(2,:)               %访问矩阵 A 第 2 行的数据
ans =
    0.7382    0.9355    0.8936    0.8132
>> A(2,:)=0              %矩阵 A 的第 2 行数据赋为 0 值
A =
    0.9218    0.4057    0.4103    0.3529
         0         0         0         0
    0.1763    0.9169    0.0579    0.0099
>> A(:,3)=1              %矩阵 A 的第 3 列数据赋为 1 值
A =
    0.9218    0.4057    1.0000    0.3529
         0         0    1.0000         0
    0.1763    0.9169    1.0000    0.0099
>> A(:,2)=[];            %删除矩阵 A 中第 2 列的数据
>> A
A =
    0.9218    1.0000    0.3529
         0    1.0000         0
    0.1763    1.0000    0.0099
>> A(2,:)=[];            %删除矩阵 A 中第 2 行的数据
A =
    0.9218    1.0000    0.3529
    0.1763    1.0000    0.0099
```

4. 全部元素的访问

矩阵全部元素访问主要还是基于下标和单下标访问机制，只是在访问的时候以符号“:”代替所有元素，例如对于二维矩阵 A 的访问， $A(:,:)$ 以矩阵形式被访问， $A(:)$ 以按列排列形式访问。具体见例 3.15。

【例 3.15】全部元素的访问。

```
>> A=rand(2,3)
A =
    0.6813    0.8318    0.7095
    0.3795    0.5028    0.4289
>> A(:,:)               %以矩阵形式访问 A 中所有元素
ans =
    0.6813    0.8318    0.7095
    0.3795    0.5028    0.4289
>> A(:)                 %以列形式访问 A 中所有元素
ans =
    0.6813
    0.3795
    0.8318
    0.5028
    0.7095
    0.4289
```

5. 对角线元素的访问

对角线元素的访问主要通过 `diag()` 函数来实现，在前面的章节中已介绍过 `diag()` 函数可用于生成对角矩阵，同时也可用于访问指定对角线上的元素，但不可利用此函数修改矩阵对角线上的值。

`diag()` 函数的用法如下。

- `diag(A)`: 用于访问矩阵 A 主对角线上的元素。
- `diag(A,k)`: 用于访问矩阵 A 第 k 条对角线上的元素，其中主对角线对应 k 为 0，主对角线以上的对角线对应的 k 依次加 1，主对角线以下的对角线对应的 k 依次减 1。

【例 3.16】对角线元素的访问。

```
>> A=rand(3,3)
A =
    0.3046    0.6822    0.1509
    0.1897    0.3028    0.6979
    0.1934    0.5417    0.3784
>> diag(A)                                %访问矩阵 A 主对角线上的元素
ans =
    0.3046
    0.3028
    0.3784
>> diag(A,1)                              %访问矩阵 A 主对角线以上一条对角线的元素
ans =
    0.6822
    0.6979
>> diag(A,-1)                             %访问矩阵 A 主对角线以下一条对角线的元素
ans =
    0.1897
    0.5417
```

6. end 在矩阵元素访问中的使用

`end` 在矩阵中代表最后一个元素，可以是一行的最后、一列的最后或者整个矩阵最后的一个元素，同时结合冒号的使用，可以访问从某个元素开始到最后的元素。使用 `end` 访问矩阵中的数据，可以不需要矩阵具体的大小，当然在知道矩阵具体尺寸的时候可以通过最后一个元素的下标来访问，效果是一样的。

【例 3.17】end 在矩阵元素访问中的使用。

```
>> A=randn(2,3)
A =
   -0.4326    0.1253   -1.1465
   -1.6656    0.2877    1.1909
>> A(2,2:end)                             %访问矩阵 A 第 2 行从第 2 列到最后一列的元素
ans =
    0.2877    1.1909
>> A(end,3)                              %访问矩阵 A 第 3 列的最后一个元素
ans =
    1.1909
>> A(end,end)                             %按双下标法访问矩阵 A 的最后一个元素
ans =
    1.1909
>> A(end)                                 %按单下标法访问矩阵 A 的最后一个元素
ans =
    1.1909
>> A(2,2:size(A,2))                       %计算出矩阵 A 的列数，然后读取矩阵 A 第 2 行，从第 2 列到最后一列的元素
ans =
    0.2877    1.1909
```

7. Find()函数在矩阵元素访问中的使用

`Find()` 函数可用于查找访问矩阵中满足一定条件的元素，用法如下。

- `indices=find(X)`: 用于查找矩阵 X 中的非零元素, 返回矩阵 X 中非零元素的线性索引。
- `indices=find(X,k)`: 用于查找矩阵 X 中的非零元素, 返回矩阵 X 中前 k 个非零元素的线性索引。
- `indices=find(X,k,'last')`: 用于查找矩阵 X 中的非零元素, 返回矩阵 X 中后 k 个非零元素的线性索引。
- `[row,col]=find(X)`: 用于查找矩阵 X 中的非零元素, 返回矩阵 X 中非零元素的行列下标。
- `[row,col]=find(X,k)`: 用于查找矩阵 X 中的非零元素, 返回矩阵 X 中前 k 个非零元素的行列下标。
- `[row,col]=find(X,k,'last')`: 用于查找矩阵 X 中的非零元素, 返回矩阵 X 中后 k 个非零元素的行列下标。

上述非零元素的查找访问, 在稀疏矩阵的操作中使用得比较多。同时可以通过矩阵的逻辑表达式, 查找符合一定条件的矩阵元素。具体 `find()` 函数的使用, 见例 3.16。

【例 3.18】 `find()` 函数在矩阵元素访问中的使用。

```
X = [1 0 4; -3 0 0; 0 8 6]
indices = find(X)           %返回矩阵 x 中大于 0 的元素的线性索引
X =
     1     0     4
    -3     0     0
     0     8     6
indices =
     1
     2
     6
     7
     9
>> [row,col]=find(X)       %返回矩阵 x 中大于零的元素的行列下标
row =
     1
     2
     3
     1
     3
col =
     1
     1
     2
     3
     3
>> indices = find(X,2)     %返回矩阵 x 中前两个大于零的元素的线性索引
indices =
     1
     2
>> [row,col]=find(X,2,'last') %返回矩阵 x 中后两个大于零的元素的行列下标
row =
     1
     3
col =
     3
     3
>> indices=find(X>2)      %返回矩阵 x 中大于 2 的元素的线性索引
indices =
     6
     7
     9
>> indices=find(X==8)     %返回矩阵 x 中等于 8 的元素的线性索引
```

```
indices =  
    6  
>> indices=find(X>8)           %返回矩阵 x 中大于 8 的元素的线性索引，矩阵 x 中不存在，返回空矩阵  
indices =  
    Empty matrix: 0-by-1
```

3.4 矩阵及数组的简单运算

矩阵运算是 MATLAB 软件操作的核心，MATLAB 中大部分的运算都是基于矩阵操作的。本节将向大家介绍矩阵及数组的简单运算，包括最基本的矩阵及数组运算的函数和代数运算。

矩阵在形式上同二维数组，但是数组运算是基于数组内每个元素的计算，即点运算，而矩阵运算是整体的运算，多用于数学概念上，采用线性代数的运算方式，因而在一些计算中矩阵与数组有差异。在矩阵及数组的代数运算部分要注意区别矩阵与数组的差异。

3.4.1 基本函数

本节将具体介绍一些与矩阵操作密切相关的函数，包括矩阵连接、变形、大小的改变等。

1. 连接函数

矩阵的连接包括水平方向左右连接和垂直方向上下连接，水平连接的矩阵需要具有相同的行数，垂直连接的矩阵需要具有相同的列数。而矩阵连接可以通过中括号“[]”或连接函数实现。

- [A;B]：用于垂直方向连接具有相同列数的矩阵 A 和 B。
- [A B]或[A,B]：用于水平方向连接具有相同行数的矩阵 A 和 B。
- cat(dim,A,B)：在指定维数上连接矩阵 A 和 B，其中，cat(1,A,B)相当于[A;B]，cat(2,A,B)相当于[A,B]和[A B]。
- horzcat(A,B)：用于水平方向连接矩阵 A 和 B。
- vertcat(A,B)：用于垂直方向连接矩阵 A 和 B。

【例 3.19】 矩阵连接操作的实现。

```
>> A=magic(3);  
>> B=ones(3);  
>> [A B]           %水平方向连接矩阵  
ans =  
    8     1     6     1     1     1  
    3     5     7     1     1     1  
    4     9     2     1     1     1  
>> [A,B]           %水平方向连接矩阵  
ans =  
    8     1     6     1     1     1  
    3     5     7     1     1     1  
    4     9     2     1     1     1  
>> [A;B]           %垂直方向连接矩阵  
ans =  
    8     1     6  
    3     5     7  
    4     9     2  
    1     1     1  
    1     1     1  
    1     1     1  
>> cat(1,A,B)       %垂直方向连接矩阵  
ans =  
    8     1     6  
    3     5     7
```

```

      4      9      2
      1      1      1
      1      1      1
      1      1      1
>> cat(2,A,B)      %水平方向连接矩阵
ans =
      8      1      6      1      1      1
      3      5      7      1      1      1
      4      9      2      1      1      1
>> horzcat(A,B)      %水平方向连接矩阵
ans =
      8      1      6      1      1      1
      3      5      7      1      1      1
      4      9      2      1      1      1
>> vertcat(A,B)      %垂直方向连接矩阵
ans =
      8      1      6
      3      5      7
      4      9      2
      1      1      1
      1      1      1
      1      1      1

```

2. 翻转函数

MATLAB 提供了一些函数可快速实现矩阵方向的变化, 通过翻转生成新的矩阵或数据。下面详细介绍主要函数和用法:

- `fliplr(A)`: 用于矩阵 A 的左右翻转, 不适用于二维以上矩阵或数组。
- `flipud(A)`: 用于矩阵 A 的上下翻转, 不适用于二维以上矩阵或数组。
- `rot90(A)`: 用于矩阵 A 逆时针 90 度翻转, `rot90(A,k)`可按逆时针方向 $90 \times k$ 度旋转矩阵 A。
- `transpose(A)`: 用于返回转置后的矩阵 A, 同 `A'`。

【例 3.20】 矩阵翻转的实现。

```

>> A=magic(3)
A =
      8      1      6
      3      5      7
      4      9      2
>> fliplr(A)      %矩阵 A 的左右翻转
ans =
      6      1      8
      7      5      3
      2      9      4
>> flipud(A)      %矩阵 A 的上下翻转
ans =
      4      9      2
      3      5      7
      8      1      6
>> rot90(A)      %矩阵 A 逆时针方向翻转 90 度
ans =
      6      7      2
      1      5      9
      8      3      4
>> rot90(A,3)      %矩阵 A 逆时针方向翻转 270 度
ans =
      4      3      8
      9      5      1
      2      7      6
>> rot90(A,-1)      %矩阵 A 顺时针方向翻转 90 度
ans =

```

```

4     3     8
9     5     1
2     7     6
>> transpose(A)           %矩阵 A 的转置
ans =
8     3     4
1     5     9
6     7     2
>> A'
ans =
8     3     4
1     5     9
6     7     2

```

3. 改变矩阵大小

矩阵的大小即矩阵的行列数。在 MATLAB 7.0 中矩阵大小的改变可以通过添加、删除、拼接元素的方法来实现,同时可以通过函数重现排列、复制矩阵,达到改变矩阵大小的目的。

本书前面章节中介绍的数组编辑窗口,通过界面操作可以实现在矩阵或数组中快速插入、删除指定行列的元素,操作类似于一般的 Excel 操作,这里主要介绍通过命令行的方式实现矩阵元素的删除等操作,改变矩阵的大小。

【例 3.21】矩阵行列的添加、删除。

```

>> A=magic(3)             %创建 3×3 的魔术阵
A =
8     1     6
3     5     7
4     9     2
>> A(:,4)=1               %在矩阵 A 中添加第 4 列元素,并赋值为 1
A =
8     1     6     1
3     5     7     1
4     9     2     1
>> A(5,:)=3               %在矩阵 A 中添加第 5 行数据并赋值为 3,第 4 行数据自动赋值为 0
A =
8     1     6     1
3     5     7     1
4     9     2     1
0     0     0     0
3     3     3     3
%添加第 6 行第 5 列元素,并赋值为 2,其余新添加的第 6 行和第 5 列的元素自动赋值为 0
>> A(6,5)=2
A =
8     1     6     1     0
3     5     7     1     0
4     9     2     1     0
0     0     0     0     0
3     3     3     3     0
0     0     0     0     2
>> A(2,:)=[]              %删除矩阵 A 中第 2 行数据
A =
8     1     6     1     0
4     9     2     1     0
0     0     0     0     0
3     3     3     3     0
0     0     0     0     2
>> A(:,3:5)=[]            %删除矩阵 A 中第 3 列到第 5 列的元素
A =
8     1
4     9
0     0

```



```

3      3
0      0
>> A(2,1)=[];           %不能以行列下标的访问机制单独删除矩阵中的一个元素，提示出错
??? Indexed empty matrix assignment is not allowed.
%以线性索引的方法，可以删除矩阵中单个元素，但返回的矩阵 A 将是按线性索引排列的一行向量
>> A(2)=[]
A =
8      0      3      0      1      9      0      3      0

```

与数组编辑窗口矩阵元素的添加操作比较，通过命令行方式改变矩阵大小无法简单实现在矩阵中间行、列插入一行或一列新的初始值为 0 的元素的。在命令行中需要使用矩阵的拼接才可实现，但在实际应用中需要代码实现而无法通过界面实现的时候，仍然会遇到这一问题。下面的例子将演示在矩阵内部插入新元素。

【例 3.22】在矩阵内部插入新的行、列。

```

>> A=rand(2,3) %产生 2×3 的随机矩阵 A
A =
0.6449    0.6602    0.2897
0.8180    0.3420    0.3412
%在矩阵 A 的第 2 行插入元素值为 1、2、3 的新的一行数据，返回矩阵 B
>> B=[A(1,:); [ 1 2 3]; A(2,:)]
B =
0.6449    0.6602    0.2897
1.0000    2.0000    3.0000
0.8180    0.3420    0.3412
%在矩阵 A 的第 2 列插入元素值为 1、2 的新的一列数据，返回矩阵 C
>> C=[A(:,1) [1;2] A(:,2:end)]
C =
0.6449    1.0000    0.6602    0.2897
0.8180    2.0000    0.3420    0.3412

```

在 MATLAB 中提供了函数可以对矩阵大小进行重排，在矩阵元素个数不变的情况下，调整矩阵各维数据的大小，而矩阵的重排顺序是按列的顺序重排，即线性索引的顺序。主要通过函数 reshape 实现，下面介绍其用法。

- $B = \text{reshape}(A, m, n)$: 用于重新排列矩阵 A，返回大小为 $m \times n$ 的矩阵 B，矩阵 A 的元素个数需要等于 $m \times n$ 。
- $B = \text{reshape}(A, m, n, p, \dots)$: 用于重新排列矩阵 A，返回大小为 $m \times n \times p \dots$ 的矩阵 B，矩阵 A 的元素个数需要等于 $m \times n \times p \dots$ ，用于高维数组的重排。
- $B = \text{reshape}(A, \dots, [], \dots)$: 重排矩阵 A，其中一维的大小可以使用默认值。
- $B = \text{reshape}(A, \text{size})$: 根据 size 函数得出的矩阵大小值重排矩阵 A。

【例 3.23】通过 reshape() 函数实现矩阵的重排。

```

>> A=[1:12]           %生成向量 A，共 12 个元素
A =
1      2      3      4      5      6      7      8      9     10     11     12
>> B=reshape(A, 3, 4) %重新排列矩阵 A，返回大小为 3×4 的矩阵 B
B =
1      4      7     10
2      5      8     11
3      6      9     12
>> C=reshape(A, 2, []) %重新排列矩阵 A，其中第二维大小设置为默认值，返回大小为 2×6 的矩阵 C
C =
1      3      5      7      9     11
2      4      6      8     10     12
>> D=reshape(A, size(C)) %重新排列矩阵 A，返回与矩阵 C 具有相同大小的矩阵 D
D =
1      3      5      7      9     11
2      4      6      8     10     12

```

MATLAB 7.0 提供了 `repmat()` 函数复制矩阵。矩阵的复制在矩阵运算中具有较大的用处，因为一般的矩阵加、减等运算要求矩阵具有相同的大小。而如果遇到一个矩阵每列（行）需要都加上或减去一个数值时，即矩阵与向量的操作，不同大小的矩阵无法相加减。如果通过循环，基于矩阵中每列（行）相加减，算法的效率不高，特别是当需要计算的矩阵很大时，对算法运行时间影响很大，而函数 `repmat()` 通过复制矩阵，可以向量生成与需要加、减的矩阵相同的矩阵，进行矩阵间的运算，算法效率较高，具体见例 3.22。`repmat()` 函数的用法如下。

- $B = \text{repmat}(A, m, n)$: 以 A 为重复单元，把 A 看做整体，返回 $m \times n$ 个 A 组成的矩阵 B 。
- $B = \text{repmat}(A, n)$: 以 A 为重复单元，把 A 看做整体，返回 $n \times n$ 个 A 组成的矩阵 B 。

【例 3.24】 `repmat()` 函数的使用。

```
>> B=repmat(eye(2),1,2)    %以 2 阶单位阵为整体，重复生成 1×2 个 2 阶单位阵构成的矩阵 B
B =
     1     0     1     0
     0     1     0     1
>> B=repmat(eye(2),2,2)    %以 2 阶单位阵为整体，重复生成 2×2 个 2 阶单位阵构成的矩阵 B
B =
     1     0     1     0
     0     1     0     1
     1     0     1     0
     0     1     0     1
>> B=repmat([1:5],1,2)    %在行方向，生成数列 1:5 重复排列两次的矩阵 B
B =
     1     2     3     4     5     1     2     3     4     5
>> B=repmat([1:5],2,1)    %在列方向，生成数列 1:5 重复排列两次的矩阵 B
B =
     1     2     3     4     5
     1     2     3     4     5
%以下代码用于计算随机矩阵 A 与其每一行均值的差
>> A=rand(3,3)
A =
    0.5341    0.8385    0.7027
    0.7271    0.5681    0.5466
    0.3093    0.3704    0.4449
>> B=A-repmat(mean(A,2),1,3)
B =
   -0.1577    0.1467    0.0110
    0.1132   -0.0458   -0.0673
   -0.0656   -0.0044    0.0700
```

4. 其他一些常用函数

下面介绍一些在矩阵操作中经常会用到的一些函数。

函数 `unique` 可用于去除矩阵中重复元素，其具体用法如下。

- $B = \text{unique}(A)$: 去除矩阵 A 中的重复元素，返回无重复元素的 A 到新变量 B ， B 以向量形式存在，并按从小到大的顺序排列 A 中无重复的元素。
- $[B, m, n] = \text{unique}(A)$: 去除 A 中重复元素，返回无重复元素的 A 到新向量 B ， m 为 B 在 A 中的线性索引值，即 $b = A(m)$ ， n 为 A 在 B 中的索引值，即 $A = b(n)$ 。
- $B = \text{unique}(A, 'rows')$: 去除矩阵 A 中的重复行，返回新矩阵 B ，其中矩阵 B 中行的排列是按照第一列元素从小到大排列，如果第一列元素相同，则依次比较后面的列。
- $[B, m, n] = \text{unique}(A, 'rows')$: 去除矩阵 A 中的重复行，返回新矩阵 B ， m 为矩阵 B 在 A 中对应的行索引，即 $B = A(m,:)$ ， n 为矩阵 A 在 B 中对应的行索引，即 $A = B(n,:)$ 。

【例 3.25】 函数 `unique()` 在矩阵操作中的使用。

```
>> A=[1 2 3 4;3 4 5 6;1 2 3 4]
A =
```

```

1     2     3     4
3     4     5     6
1     2     3     4
>> B=unique(A)           %获取矩阵 A 中无重复元素的新向量 B，B 中元素按从小到大的顺序排列
B =
1
2
3
4
5
6
>> [B,i,j]=unique(A)     %获取矩阵 A 中无重复元素的新向量 B，B 中元素按从小到大的顺序排列，并返回 B
中元素在 A 中的索引值 i 和 A 中元素在 B 中的索引值 j
B =
1
2
3
4
5
6
i =
3
6
9
12
8
11
j =
1
3
1
2
4
2
3
5
3
4
6
4
>> B=unique(A, 'rows')   %获取矩阵 A 中无重复行的新矩阵 B，B 中行按从小到大的顺序排列
B =
1     2     3     4
3     4     5     6
>> [B,i,j]=unique(A, 'rows') %获取矩阵 A 中无重复行的新矩阵 B，B 中行按从小到大的顺序排列，并返回 B
中元素在 A 中的行索引值 i 和 A 中元素在 B 中的行索引值 j
B =
1     2     3     4
3     4     5     6
i =
3
2
j =
1
2
1

```

在上面的操作中，有时需要用到矩阵行列下标，有时用到矩阵线性下标索引，对于小型矩阵，可以简单看出行列下标和线性索引间的对应关系，但是复杂矩阵下标的转换就比较复杂，特别是线性索引转换为行列下标的时候比较麻烦，可以使用 MATLAB 提供的函数 `sub2ind` 和 `ind2sub` 实现，同时函数适用于高维数组。下面具体介绍函数的用法。

- `indces=sub2ind(size, i, j)`: 用于把矩阵的行列下标转换为线性索引下标, 其中 `size` 为需要转换的矩阵的维数, `i` 和 `j` 分别为需要转换的行、列下标值, `indces` 返回大小为 `size` 的矩阵 `i` 行 `j` 列对应的线性索引值。
- `[i,j]=ind2sub(size, indces)`: 用于将矩阵的线性索引下标转换为行列下标, 其中 `size` 为需要转换的矩阵的维数, `indces` 为需要转换的线性索引下标值, 函数返回的 `i` 和 `j` 分别为 `indces` 对应的行、列下标值。

【例 3.26】 矩阵下标转换的实现。

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> sub2ind(size(A),1,2) %返回矩阵 A 中第 1 行第 2 列元素的线性索引值
ans =
     4
>> [i,j]= ind2sub(size(A),4) %返回矩阵 A 线性索引值 4 对应的矩阵的行列坐标
i =
     1
j =
     2
>> A(1,2) %输出矩阵 A 中第 1 行第 2 列元素值, 对应线性索引为 4, 验证下标转换
ans =
     1
>> A(4) %按线性索引方式输出矩阵 A 第 1 行第 2 列元素, 验证下标转换
ans =
     1
```

3.4.2 加减运算

矩阵的加减运算要求相加减的矩阵有相同的维数, 即相同的行列数, 与线性代数中的运算法则是一致的, 用法如下。

- $C=A+B$: 用于矩阵间的加法运算。
- $C=A-B$: 用于矩阵间的减法运算。

如果在特殊情况下, A 和 B 中有一个变量为标量, 则可不要求 A 、 B 具有相同维数, 相当于矩阵中的每个元素与标量进行相加减的操作。

【例 3.27】 矩阵的加减运算。

```
>> A=[ 1 3 5;6 8 4;2 5 8]
A =
     1     3     5
     6     8     4
     2     5     8
>> B=[2 3 5;3 4 6;7 2 5]
B =
     2     3     5
     3     4     6
     7     2     5
>> A+B %矩阵的加法运算
ans =
     3     6    10
     9    12    10
     9     7    13
>> A-B %矩阵的减法运算
ans =
    -1     0     0
     3     4    -2
```

```

-5    3    3
>> A+2           %矩阵与标量的加法运算
ans =
    3    5    7
    8   10    6
    4    7   10
>> A-3           %矩阵与标量的减法运算
ans =
   -2    0    2
    3    5    1
   -1    2    5

```

3.4.3 乘法运算

矩阵与数组在乘法运算中有所区别，矩阵的乘法是线性代数中常用的运算，要求被乘矩阵的列数等于相乘矩阵的行数。而数组的乘法是点乘运算，即数组具有相同下标的元素相乘，运算时在一般乘法运算的“*”前加上“.”，需要两数组具有相同的维数。乘法的运算用法如下。

- $C=A*B$ ：用于矩阵的乘法，要求矩阵 A 的列数等于矩阵 B 的行数。
- $C=A.*B$ ：用于数组的乘法，要求矩阵 A 和 B 具有相同的大小。

注意，如果 A 和 B 中有一个为标量，则不需要遵守上述规则，标量与矩阵中每个元素相乘。

【例 3.28】 矩阵及数组的乘法运算。

```

>> A=[1 2 3;4 5 6]           %生成 2×3 的矩阵 A
A =
    1    2    3
    4    5    6
>> B=[3 2;1 2;4 5]           %生成 3×2 的矩阵 B
B =
    3    2
    1    2
    4    5
>> A*B                         %矩阵 A 和 B 相乘
ans =
   17   21
   41   48
>> C=B'                         %求取 B 的转置矩阵 C，大小为 2×3
C =
    3    1    4
    2    2    5
>> A*C                         %矩阵 A 和 C 相乘，A 的列数和 C 的行数不相等，程序出错
??? Error using ==> mtimes
Inner matrix dimensions must agree.
>> A.*C                         %具有相同维数的数组 A 和 C 进行点乘
ans =
    3    2   12
    8   10   30
>> A.*2                         %矩阵与标量的点乘
ans =
    2    4    6
    8   10   12
>> A*2                         %矩阵与标量的数乘
ans =
    2    4    6
    8   10   12

```

3.4.4 除法运算

矩阵的除法与数组的除法有一定差异，其中矩阵的除法对应与线性代数中的逆运算相关，而

数组的除法是点除，是数组中相同下标元素的相除。同时除法运算又分为左除和右除。

1. 矩阵的除法

- $A \setminus B$ ：矩阵的左除，如果 A 为方阵，即 $n \times n$ 阶矩阵，可用于计算方程 $Ax=B$ 的解，等效于 $\text{inv}(A)*B$ 。
- B/A ：矩阵的右除，用于计算方程 $xA=B$ 的解，等效于 $B*\text{inv}(A)$ 。

【例 3.29】 矩阵的除法运算。

```
>> A=[2 1;3 2]
A =
     2     1
     3     2
>> B=[8 4;13 7]
B =
     8     4
    13     7
>> x=A\B                                %矩阵的左除运算，等效于计算方程 Ax=B 的解
x =
    3.0000    1.0000
    2.0000    2.0000
>> A*x                                    %验证矩阵的左除运算，A*x 值为矩阵 B
ans =
     8     4
    13     7
>> x=B/A                                    %矩阵的右除运算，等效于计算方程 Ax=B 的解
x =
     4     0
     5     1
>> x*A                                    %验证矩阵的右除运算，x*A 值为矩阵 B
ans =
     8     4
    13     7
>> A/2                                    %矩阵与标量的右除运算
ans =
    1.0000    0.5000
    1.5000    1.0000
>> 2\A                                    %标量与矩阵的左除运算
ans =
    1.0000    0.5000
    1.5000    1.0000
```

2. 数组的除法

数组的除法为点除，是对应下标元素的除法，因此要求进行除法的数组具有相同的维数，具体的用法如下。

- $A ./ B$ ：数组的左除，即 $A(i,j)/B(i,j)$ 。
- $A . \setminus B$ ：数组的右除，即 $B(i,j)/A(i,j)$ 。

注意，如果 A 是标量，数组的左除 $A ./ B$ 即 A 除以矩阵数组 B 的每一个元素，右除 $A . \setminus B$ 即矩阵数组 B 的每个元素除以标量 A ；如果 B 是标量，则数组的左除 $A ./ B$ 即矩阵数组 A 的每个元素除以 B ，右除 $A . \setminus B$ 即标量 B 除以矩阵数组 A 的每一个元素。

【例 3.30】 数组的除法运算。

```
>> A=[2 4 6;8 9 4]
A =
     2     4     6
     8     9     4
>> B=[2 2 3;4 3 1]
B =
```

```

    2     2     3
    4     3     1
>> A./B %数组的左除
ans =
    1     2     2
    2     3     4
>> A.\B %数组的右除
ans =
    1.0000    0.5000    0.5000
    0.5000    0.3333    0.2500
>> A./2 %数组与标量的左除
ans =
    1.0000    2.0000    3.0000
    4.0000    4.5000    2.0000
>> A.\2 %数组与标量的右除
ans =
    1.0000    0.5000    0.3333
    0.2500    0.2222    0.5000
>> 2./B %标量与数组的左除
ans =
    1.0000    1.0000    0.6667
    0.5000    0.6667    2.0000
>> 2.\B %标量与数组的右除
ans =
    1.0000    1.0000    1.5000
    2.0000    1.5000    0.5000

```

3.4.5 乘方运算

矩阵与数组在乘方运算中也有不同，矩阵的乘方相当于多个矩阵相乘，而数组的乘方即数组中每个元素的乘方运算。具体用法如下。

- 矩阵的乘方运算： $C=A^B$ 。其中， A 需为方阵，即矩阵的行列数相等， B 为标量。 A^B 的运算即相当于 B 个矩阵 A 相乘。
- 数组的乘方运算： $C=A.^B$ 。其中，当 A 、 B 都为数组时，需大小相等； $C(i,j)$ 的计算结果即为 $A(i,j)$ 的 $B(i,j)$ 次方；如果 A 为标量， $A.^B$ 的运算即相当于分别对元素 A 做 $B(i,j)$ 次方；如果 B 为标量， $A.^B$ 的运算即相当于对数组 A 中的每个元素做 B 阶乘方运算。

【例 3.31】乘方运算。

```

>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> A^2 %矩阵（方阵）的乘方运算
ans =
    91    67    67
    67    91    67
    67    67    91
>> A=[1 2 3]
A =
     1     2     3
>> A^2 %不支持不为方阵的矩阵的乘方运算
??? Error using ==> mpower
Matrix must be square.
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2

```

```
>> B=[1 2 3;1 2 3;1 2 3]
B =
     1     2     3
     1     2     3
     1     2     3
>> A.^B                                %数组的乘方运算
ans =
     8     1    216
     3    25    343
     4    81     8
>> A.^2                                %数组与标量的乘方运算
ans =
    64     1    36
     9    25    49
    16    81     4
>> 2.^B                                %标量与数组的乘方运算
ans =
     2     4     8
     2     4     8
     2     4     8
```

3.5 矩阵的特殊运算

矩阵的特殊操作主要指线性代数中矩阵的行列式、求逆、特征值、秩运算，在本节中将详细介绍矩阵这些特殊运算的实现。

3.5.1 行列式运算

行列式是线性代数运算中的重要工具，在 MATLAB 7.0 中提供函数 `det` 用于计算矩阵的行列式。如果矩阵为方阵，则其存在行列式，可通过函数 `det` 计算出矩阵的行列式值，此值为一标量。

【例 3.32】 矩阵的行列式计算。

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2
>> det(A)                             %矩阵 A 的行列式计算
ans =
   -360
```

3.5.2 逆运算

在线性代数中，若矩阵 A 是方阵，且为非奇异阵，即行列式值不为 0，存在矩阵 x 使 $Ax=I$ 和 $xA=I$ ， x 称为矩阵 A 的逆矩阵，记做 A^{-1} 。在 MATLAB 7.0 中用函数 `inv()` 来计算矩阵的逆运算。

【例 3.33】 矩阵的逆运算。

```
>> A=magic(3);
>> X=inv(A)                            %矩阵 A 求逆运算
X =
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028
>> X*A                                  %验证矩阵 A 求逆运算
ans =
    1.0000   -0.0000    0.0000
    0.0000    1.0000   -0.0000
```



```

0      0.0000      1.0000
>> A*X                                %验证矩阵 A 求逆运算
ans =
    1.0000    0.0000   -0.0000
   -0.0000    1.0000    0.0000
    0.0000    0.0000    1.0000
>> A^-1                                %通过乘方运算完成矩阵 A 求逆运算
ans =
    0.1472   -0.1444    0.0639
   -0.0611    0.0222    0.1056
   -0.0194    0.1889   -0.1028

```

3.5.3 秩运算

矩阵中线性无关的行数与列数称为矩阵的秩。在 MATLAB 7.0 中，函数 rank() 用于求矩阵的秩。

【例 3.34】 矩阵的秩运算。

```

>> A=magic(3);
>> r=rank(A)                            %矩阵秩的计算
r =
    3

```

3.5.4 特征值运算

在 MATLAB 7.0 中使用函数 eig 计算矩阵的特征值，函数的用法如下。

- e=eig(x): 其中 e 是由特征值组成的列向量，x 是输入的方阵。
- [v,d]=eig(x): 方阵 x 的全部特征值，构成对角阵 d，其对角线即为特征值，v 是一个与 x 相同大小的矩阵，每一列是矩阵 x 的一个特征值所对应的特征向量。

【例 3.35】 矩阵的特征值运算。

```

>> x=magic(3);
>> e=eig(x)                            %矩阵 x 的特征值计算
e =
    15.0000
     4.8990
    -4.8990
>> [v,d]=eig(x)                        %矩阵 x 的特征值和特征向量计算
v =
   -0.5774   -0.8131   -0.3416
   -0.5774    0.4714   -0.4714
   -0.5774    0.3416    0.8131
d =
    15.0000         0         0
         0     4.8990         0
         0         0    -4.8990

```

3.6 数组的特殊运算

在 MATLAB 中，数组的特殊运算主要包括关系运算、逻辑运算和集合运算，与一般的矩阵运算不同的是，这些运算都是基于数组中每个元素进行的。下面将详细介绍这些运算关系的使用。

3.6.1 关系运算

MATLAB 中的关系运算主要用于判断数组的大小关系，关系成立返回“1”，关系不成立则返回“0”。MATLAB 7.0 中提供的关系符有大于(>)，小于(<)，大于等于(>=)，小于等于(<=)，

【例 3.36】数组的关系运算。

```
>> A=magic(3)
A =
     8     1     6
     3     5     7
     4     9     2

>> B= 1 + fix(10* rand(3))
B =
    10     5     5
     3     9     1
     7     8     9

>> C=A>B                                     %判断数组 A 中的元素是否大于数组 B 中的相应元素
C =
     0     0     1
     0     0     1
     0     1     0

>> C=gt(A,B)                                %通过函数判断数组 A 中的元素是否大于数组 B 中的相应元素
C =
     0     0     1
     0     0     1
     0     1     0

>> C=A>2                                     %判断数组 A 中的元素是否大于 2
C =
     1     0     1
     1     1     1
     1     1     0

>> [I,J]=find(A<3)                           %查找数组 A 中小于 3 的元素，I、J 分别返回数组的行列坐标
I =
     1
     3

J =
     2
     3
```

[illegible]

```

0     0     0
0     0     1
0     0     0

```

(3) 数组的小于运算, 关系表达式为 $C=A<B$ 或 $C=(A<B)$, 对应的函数为 $C=lt(A,B)$, 数组 C 返回用于判断数组 A 中各元素是否小于数组 B 中相应位置元素的 0、1 逻辑数组, 1 代表关系成立, 0 代表关系不成立。

```

>> C=A<B                                %判断数组 A 中的元素是否小于数组 B 中的相应元素
C =
     1     1     0
     0     1     0
     1     0     1

>> C=A<5                                %判断数组 A 中的元素是否小于 5
C =
     0     1     0
     1     0     0
     1     0     1

>> C=lt(A,5)                            %通过函数判断数组 A 中的元素是否小于 5
C =
     0     1     0
     1     0     0
     1     0     1

```

(4) 数组的小于等于运算, 关系表达式为 $C=A\leq B$ 或 $C=(A\leq B)$, 对应的函数为 $C=le(A,B)$, 数组 C 返回用于判断数组 A 中各元素是否小于等于数组 B 中相应位置元素的 0、1 逻辑数组, 1 代表关系成立, 0 代表关系不成立。

```

>> C=A<=B                               %判断数组 A 中的元素是否小于等于数组 B 中的相应元素
C =
     1     1     0
     1     1     0
     1     0     1

>> C=le(A,B)                            %通过函数判断数组 A 中的元素是否小于等于数组 B 中的相应元素
C =
     1     1     0
     1     1     0
     1     0     1

>> C=A<=2                               %通过函数判断数组 A 中的元素是否小于 2
C =
     0     1     0
     0     0     0
     0     0     1

```

(5) 数组的等于运算, 关系表达式为 $C=A==B$ 或 $C=(A==B)$, 对应的函数为 $C=ge(A,B)$, 数组 C 返回用于判断数组 A 中各元素是否大于等于数组 B 中相应位置元素的 0、1 数组, 1 代表关系成立, 0 代表关系不成立。对于数组的等于运算要与赋值运算严格的区别, 等于运算 ($==$) 判断数组中元素是否相等, 是大小的比较运算, 而赋值运算 ($=$) 是把具体的值赋值给相应的变量。

```

>> C=A==B                               %判断数组 A 中的元素是否等于数组 B 中的相应元素
C =
     0     0     0
     1     0     0
     0     0     0

>> C=eq(A,B)                            %通过函数判断数组 A 中的元素是否等于数组 B 中的相应元素
C =
     0     0     0
     1     0     0
     0     0     0

>> C=A==2                               %通过函数判断数组 A 中的元素是否等于 2
C =
     0     0     0
     0     0     0
     0     0     1

```

```
>> A=2                                %赋值运算，使变量 A 的值为 2
A =
    2
```

(6) 数组的不等运算，关系表达式为 $C=A \sim B$ 或 $C=(A \sim B)$ ，对应的函数为 $C = \text{ne}(A,B)$ ，数组 C 返回用于判断数组 A 中各元素是否不等于数组 B 中相应位置元素的 0、1 数组，1 代表关系成立，0 代表关系不成立。

```
>> C=A~B                                %判断数组 A 中的元素是否不等于数组 B 中的相应元素
C =
     1     1     1
     1     1     1
     1     1     1

>> C=ne(A,B)                            %通过函数判断数组 A 中的元素是否不等于数组 B 中的元素
C =
     1     1     1
     1     1     1
     1     1     1
```

3.6.2 逻辑运算

MATLAB 7.0 中提供的逻辑运算主要有与运算 (&)、或运算 (|)、非运算 (~)、异或运算 (xor)、快速逻辑与运算 (&&)、快速逻辑或运算 (||) 和逻辑函数 all、any。下面具体介绍各逻辑运算的使用。

- 与运算 (&): $C=A \& B$ ，当数组 A 、 B 相应位置上的元素都为非零元素，即返回 1，否则返回 0。
- 或运算 (|): $C=A | B$ ，当数组 A 、 B 相应位置上的元素至少一个为非零元素，即返回 1，否则返回 0。
- 非运算 (~): $C=\sim A$ ，当数组 A 相应位置上的元素都为非零元素，即返回 0，否则返回 1。
- 异或运算 (xor): $C=\text{xor}(A,B)$ ，当数组 A 、 B 相应位置上的元素一个为非零元素，一个为零，即返回 1，否则返回 0。
- 快速逻辑与运算 (&&): $C=A \&\& B$ ，其中， A 、 B 需为逻辑变量或标量，当 A 、 B 都为真或为标量时不为 0，即返回 1，否则返回 0。在进行快速逻辑与运算的时候，如果 A 已发现是零，则不去判断 B 是否为零，直接返回结果“0”；但是如果 A 为 1，即仍然要判断 B 是否为零。
- 快速逻辑或运算 (||): $C=A || B$ ， A 、 B 需为逻辑变量或标量，当 A 、 B 存在一个为真或一个标量不为 0，即返回 1，否则返回 0。快速逻辑或运算当 A 判断结果为非零后，无须继续判断 B ，即返回结果“1”；如果 A 为零，则需进一步判断 B 是否为非零。
- all(x): 判断数组 x 中的元素是否都为非零元素，是则返回“1”，否则返回“0”。当 x 为数组时，默认对列数据进行判断，即判断数组中各列元素是否都为非零元素。而如果需要判断行元素是否都为非零元素即指定判断的维数， $\text{all}(x,\text{dim})$ ，其中 $\text{dim}=2$ 用于行数据判断。
- any(x): 判断数组 x 中的元素是否存在任何一个元素为非零元素，是则返回“1”，否则返回“0”。当 x 为数组时，默认对列数据进行判断，即判断数组中各列元素是否都为非零元素。而如果需要判断行元素是否存在非零元素即指定判断的维数， $\text{any}(x,\text{dim})$ ，其中 $\text{dim}=2$ 用于行数据判断。

【例 3.37】 数组的逻辑运算。

```
>> A=eye(3)
A =
     1     0     0
     0     1     0
```

```

    0    0    1
>> B=round(rand(3))
B =
    1    0    0
    1    1    1
    1    0    1
>> C=A&B                                %数组 A、B 对应元素的逻辑与操作
C =
    1    0    0
    0    1    0
    0    0    1
>> C=A|B                                %数组 A、B 对应元素的逻辑或操作
C =
    1    0    0
    1    1    1
    1    0    1
>> C=~A                                  %数组 A 的逻辑非操作
C =
    0    1    1
    1    0    1
    1    1    0
>> C=xor(A,B)                           %数组 A、B 对应元素的逻辑异或操作
C =
    0    0    0
    1    0    1
    1    0    0
>> A && B                                %数组 A、B 的快速逻辑与操作出错
??? Operands to the || and && operators must be convertible to logical scalar values
%判断 A 是否为空矩阵, 如果 A 为空, 则不需要进行下一步判断矩阵 A 的大小
>> ~isempty(A) && size(A,2)==3
ans =
    1
>> ~isempty(A) || size(A,2)==1 %快速的逻辑或操作
ans =
    1
>> C=all(A)                              %判断数组 A 中各列元素是否均为零
C =
    0    0    0
>> C=all(B)                              %判断数组 B 中各列元素是否均为零
C =
    1    0    0
>> C=all(A,2)                            %判断数组 A 中各行元素是否均为零
C =
    0
    0
    0
>> C=all(B,2)                            %判断数组 B 中各行元素是否均为零
C =
    0
    1
    0
>> C=any(A)                              %判断数组 A 中各列元素是否存在不为零的元素
C =
    1    1    1
>> C=any(A,2)                            %判断数组 A 中各行元素是否存在不为零的元素
C =
    1
    1
    1

```

3.7 向量及其运算

在 MATLAB 中把向量看成是一维的数组,因此向量与数组或矩阵的一些操作是类似的,但同时 MATLAB 也提供了一些函数、算法用于进行向量特殊的运算。在本小节中将介绍向量的基础知识,重点介绍向量与数组或矩阵运算差异的地方。

3.7.1 向量的生成

只有一行或一列元素的数组或矩阵即为向量,向量可以看成一维的数组或矩阵。向量可分为行向量和列向量。向量的生成方法主要有命令行直接输入、用冒号生成和函数生成法,其中命令行直接输入与数组的创建方法相同,下面主要讲述利用冒号表达式建立一个向量和函数方法创建向量。

1. 利用冒号法生成向量

利用冒号法生成向量主要用于生成具有等间隔的向量,其格式为: $x_0:xstep:xend$ 。其中, x_0 为向量的初始值, $xstep$ 为步长, $xend$ 为终止值。

【例 3.38】利用冒号法生成向量。

```
>> x=2:2:10           %生成初值为 2, 终值为 10 的向量, 步长为 2
x =
     2     4     6     8    10
%生成初值为 1, 终值为 10 的向量, 步长为 2, 但终值与初值的差不是步长的整数倍, 得到的向量的%最后一个值小
于终值
>> x=1:2:10
x =
     1     3     5     7     9
>> x=1:10             %生成初值为 1, 终值为 10 的向量, 默认情况下步长为 1
x =
     1     2     3     4     5     6     7     8     9    10
```

2. 函数法生成向量

在 MATLAB 7.0 中, 提供函数 `linspace()` 和 `logspace()` 函数产生等差排列的向量。与冒号法生成向量不同的是不需要规定向量的步长, 其调用格式如下。

- `linspace(a,b,n)`: a 和 b 是生成向量的初值和终值, n 是向量中元素的个数。当 n 默认时, 默认生成 $a \sim b$ 范围内的 100 个元素。
- `logspace(a,b,n)`: 10^a 和 10^b 是生成向量的初值和终值, n 是向量中元素的个数。当 n 默认时, 默认生成 $10^a \sim 10^b$ 范围内的 50 个元素。

【例 3.39】利用函数法生成向量。

```
>> x=linspace(1,10,3)      %生成 1~10 范围内 3 个元素构成的等差向量
x =
     1.0000     5.5000    10.0000
>> x=linspace(1,10,7)      %生成 1~10 范围内 7 个元素构成的等差向量
x =
     1.0000     2.5000     4.0000     5.5000     7.0000     8.5000    10.0000
>> x=logspace(1,2,5)        %生成 10~100 范围内 5 个元素构成的等差向量
x =
    10.0000    17.7828    31.6228    56.2341   100.0000
```

3.7.2 向量的运算

向量的一般运算加、减、乘、除与数组的运算类似, 主要是按位运算, 即向量中各元素的计算。向量同样也可以进行逻辑运算和关系运算, 与数组的操作类似, 相当于一维数组的操作。本

小节将主要介绍向量有别于数组的一些函数和运算。

1. 向量运算的常用函数

1) 判断是否为向量

函数 `isvector(X)` 用于判断数据 `X` 是否为向量，如果是向量返回“1”，否则返回“0”。

【例 3.40】 向量的判断。

```
>> A=magic(3);
>> c1=isvector(A)           %判断矩阵 A 是否为向量，A 不是向量返回“0”
c1 =
    0
>> c2=isvector(A(1,:))      %判断矩阵 A 中一行数据是否为向量，是向量返回“1”
c2 =
    1
>> c3=isvector(A(:,2))      %判断矩阵 A 中一列数据是否为向量，是向量返回“1”
c3 =
    1
```

2) 向量的内积

在 MATLAB 7.0 中函数 `dot` 用于计算向量的内积，其用法如下。

```
C=dot(A,B)
```

其中，向量 `A`、`B` 需要具有相同的行数和列数。

【例 3.41】 向量的内积。

```
>> A=[1:3];
>> B=linspace(1,10,3);
>> C=dot(A,B)               %向量的内积运算
C =
    42
>> C=A*B'                   % A*B' 与向量的内积运算具有相同的结果
C =
    42
```

3) 向量的外积

在 MATLAB 7.0 中函数 `cross` 用于计算向量的外积，其用法如下。

```
C=cross(A,B)
```

其中，向量 `A`、`B` 需要具有相同的行数和列数，且行数或列数必须等于 3，矩阵 `C` 用于返回向量 `A` 和 `B` 的外积结果。

【例 3.42】 向量的外积。

```
>> A=[1:3];
>> B=linspace(1,10,3);
>> C=cross(A,B)             %向量的外积运算
C =
    3.5000   -7.0000    3.5000
```

2. 集合运算

集合运算主要用于查找两个向量元素的交集、并集、差集、异或等操作。下面具体介绍向量的集合运算。

1) 交集

向量 `A`、`B` 的交集是同时属于向量 `A` 和 `B` 的集合元素，调用的函数为 `intersect`，函数调用格式。

- `C = intersect(A,B)`: 返回向量 `A` 和 `B` 中的相同的元素，返回的结果 `C` 为相同元素按升序顺序排列的向量。
- `C = intersect(A,B,'rows')`: 返回数组 `A` 和 `B` 中具有相同的元素的行向量，返回的结果 `C` 为含有相同元素的行，如果矩阵 `A` 和 `B` 中具有多行元素含有相同的行元素，则 `C` 中返回的结

果按行元素的升序排列。

- `[C,ia,ib] = intersect(A,B)`: 返回向量 A 和 B 中的相同的元素, 返回的结果 C 为相同元素按升序顺序排列的向量; ia 和 ib 为返回的结果向量 C 在向量 A 和 B 中的线性索引, 即 $C = A(ia)$ 和 $C = B(ib)$ 。
- `[C,ia,ib] = intersect(A,B,'rows')`: 返回数组 A 和 B 中具有相同元素的行, 返回的结果 C 为相同元素按升序顺序排列的向量; ia 和 ib 为返回的结果向量 C 在向量 A 和 B 中的行索引, 即 $C = A(ia,:)$ 和 $C = B(ib,:)$ 。

【例 3.43】 向量的交集运算。

```
>> A=[1:10];
>> B=[2:3:15];
>> C = intersect(A,B)           %向量 A 和 B 的交集计算
C =
     2     5     8
>> [C,ia,ib] = intersect(A,B)   %向量 A 和 B 的交集计算, 同时返回交集元素在 A 和 B 中的索引
C =
     2     5     8
ia =
     2     5     8
ib =
     1     2     3
>> A(ia)
ans =
     2     5     8
>> B(ib)
ans =
     2     5     8
>> AA=[1 2 3;4 5 6;7 8 9];
>> BB=[2 4 5;4 5 6;7 8 1];
>> C = intersect(AA,BB,'rows')  %计算矩阵 A 和 B 中具有相同元素的行
C =
     4     5     6
%计算矩阵 A 和 B 中具有相同元素的行, 同时返回交集的行索引
>> [C,ia,ib] = intersect(AA,BB,'rows')
C =
     4     5     6
ia =
     2
ib =
     2
>> AA(2,:)
ans =
     4     5     6
>> BB(2,:)
ans =
     4     5     6
>> AA=[1 2 3;7 8 9;4 5 6];
>> BB=[2 3 4;7 8 9;4 5 6];
>> C = intersect(AA,BB,'rows')
C =
     4     5     6
     7     8     9
```

2) 并集

向量 A、B 的并集是同时属于向量 A 或 B 的集合元素, 调用的函数为 `union`, 函数调用格式如下。

- `C = union(A, B)`: 返回向量 A 和 B 所有元素, 返回的结果 C 为所有元素按升序顺序排列的

向量。

- `C = union(A, B, 'rows')`: 返回数组 A 和 B 中所有不相同的行向量。
- `[C, ia, ib] = union(A, B)`: 返回向量 A 和 B 所有元素, 返回的结果 C 为所有元素按升序顺序排列的向量, ia 和 ib 为返回的结果向量 C 在向量 A 和 B 中的线性索引, 即 C 为 A(ia)和 B(ib)元素按照升序排列的所有行元素的集合。
- `[C, ia, ib] = union(A, B, 'rows')`: 返回数组 A 和 B 中所有不相同的行向量; ia 和 ib 为返回的结果向量 C 在向量 A 和 B 中的行索引, 即 C 为 A(ia)和 B(ib)行向量元素按照升序排列的所有行元素的集合。

【例 3.44】向量的并集运算。

```
>> A=[1:10];
>> B=[2:3:15];
>> C = union(A, B)           %向量 A 和 B 的并运算, 返回向量 A 和 B 中无重复的所有元素到向量 C
C =
     1     2     3     4     5     6     7     8     9    10    11    14
>> [C, ia, ib] = union(A, B) %向量 A 和 B 的并运算, 同时返回 C 中所有元素在向量 A 和 B 中的索引
C =
     1     2     3     4     5     6     7     8     9    10    11    14
ia =
     1     3     4     6     7     9    10
ib =
     1     2     3     4     5
>> C= union(AA, BB, 'rows') %返回矩阵 AA 和 BB 的无重复的所有行元素的集合
C =
     1     2     3
     1     2     4
     4     4     4
     4     5     6
     7     8     9
%返回矩阵 AA 和 BB 的无重复的所有行元素的集合, 同时返回索引
>> [C, ia, ib] = union(AA, BB, 'rows')
C =
     1     2     3
     1     2     4
     4     4     4
     4     5     6
     7     8     9
ia =
     1
     2
ib =
     1
     2
     3
```

3) 差集

向量 A、B 的差集是属于向量 A 但不属于向量 B 的集合元素, 调用的函数为 `setdiff`, 函数调用格式如下。

- `C = setdiff(A, B)`: 向量 C 中返回属于向量 A 但不属于向量 B 的集合元素。
- `C= setdiff(A, B, 'rows')`: 返回矩阵 C 是每一行元素都属于矩阵 A 但不属于矩阵 B 的矩阵。
- `[C, i] = setdiff(A, B)`: 向量 C 中返回属于向量 A 但不属于向量 B 的集合元素, i 为 C 中元素在向量 A 中的索引, 即 $C=A(i)$ 。
- `[C, i] = setdiff(A, B, 'rows')`: 返回矩阵 C 是每一行元素都属于矩阵 A 但不属于矩阵 B 的矩阵; i 为矩阵 C 在矩阵 A 中的行索引, 即 $C=A(i,:)$ 。

【例 3.45】向量的差集运算。

```

>> A=[1:10];
B=[2:3:15];
>> C = setdiff(A, B)%向量 C 中返回属于向量 A 但不属于向量 B 的集合元素
C =
     1     3     4     6     7     9    10
%向量 C 中返回属于向量 A 但不属于向量 B 的集合元素, i 为 C 中元素在向量 A 中的索引
>> [C, i] = setdiff(A, B)
C =
     1     3     4     6     7     9    10
i =
     1     3     4     6     7     9    10
>> A(i)
ans =
     1     3     4     6     7     9    10
>> AA=[1 2 3; 4 5 6;7 8 9];
>> BB=[1 2 3; 5 5 5 ;3 2 1];
>> C = setdiff(AA, BB,'rows')%矩阵 C 中返回属于矩阵 AA 但不属于矩阵 BB 的行元素
C =
     4     5     6
     7     8     9
>> [C,i]= setdiff(AA, BB,'rows') %矩阵 C 中返回属于矩阵 AA 但不属于矩阵 BB 的行元素, i 为行索引
C =
     4     5     6
     7     8     9
i =
     2
     3

```

4) 异或

向量 A、B 的异或是属于向量 A 和 B 的并集但不属于向量 A 和 B 的交集的集合元素, 调用的函数为 setxor, 函数调用格式如下。

- C= setxor(A, B): 返回属于向量 A 和 B 的并集但不属于向量 A 和 B 的交集的集合元素。
- C = setxor(A, B, 'rows'): 属于矩阵 A 和 B 的并集但不属于矩阵 A 和 B 的交集的行元素, 即 C 中每一行元素属于矩阵 A 不属于矩阵 B 或属于矩阵 B 不属于矩阵 A。
- [C, ia, ib] = setxor(A, B): 返回属于向量 A 和 B 的并集但不属于向量 A 和 B 的交集的集合元素, ia 和 ib 为返回的结果向量 C 在向量 A 和 B 中的线性索引, 即 C 为 A(ia)和 B(ib)元素按照升序排列的所有行元素的集合。
- [C, ia, ib]= setxor(A, B, 'rows'): 属于矩阵 A 和 B 的并集但不属于矩阵 A 和 B 的交集的行元素, ia 和 ib 为返回的结果 C 在矩阵 A 和 B 中的行索引, 即 C 为 A(ia,:)和 B(ib,:)元素按照升序排列的所有行元素的集合。

【例 3.46】向量的异或运算。

```

>> A=[1:10];
B=[2:3:15];
>> C= setxor(A, B) %向量 A、B 的异或运算
C =
     1     3     4     6     7     9    10    11    14
>> [C, ia, ib] = setxor(A, B) %向量 A、B 的异或运算, 同时返回结果的索引
C =
     1     3     4     6     7     9    10    11    14
ia =
     1     3     4     6     7     9    10
ib =
     4     5
>> AA=[1 2 3;4 5 6; 7 8 9];
>> BB=[3 2 4;1 2 3;6 4 3];

```

```
>> C = setxor(AA, BB, 'rows') %基于行的矩阵 A、B 的异或运算

C =
     3     2     4
     4     5     6
     6     4     3
     7     8     9

>> [C, ia, ib]= setxor(AA, BB, 'rows') %基于行的矩阵 A、B 的异或运算，同时返回结果的行索引
C =
     3     2     4
     4     5     6
     6     4     3
     7     8     9
ia =
     2
     3
ib =
     1
     3
```

3.8 高维数组操作

除了上面介绍的矩阵、数组、向量外，MATLAB 还支持高维数组的使用，高维数组即二维以上的数组。不过一般情况下高维数组多用于存储数组，特别在图像处理中，对于一幅图像数据，多存储为高维数组。一般涉及的比较多的高维数组还是三维的，所以本节主要以三维数组的操作讲述为主。在实际操作中高维数组的数据查看使用不是很方便，同时部分函数不支持高维数组，因而在计算中我们常常还是把高维数组转换为一般的矩阵操作。今后读者如果遇到更高维的数组需要处理，建议还是转换为低维数组操作。

3.8.1 高维数组的创建

高维数组的创建方法与一般的矩阵、数组的创建方法类似，包括外部文件导入法、直接输入法和函数法，这里主要介绍后两种。

1. 直接输入法

直接输入法主要是通过给相应下标的数组元素赋值实现。三维数组在原来矩阵的行、列基础上，增加了页作为第三维数组的表示。

【例 3.47】直接输入法创建高维数组。

```
>> A(2,3,1:4)=[2 4 6 8] %数组 A 第 1~4 页的第 2 行第 3 列的元素分别赋值为 2、4、6、8
A(:, :, 1) =
     0     0     0
     0     0     2
A(:, :, 2) =
     0     0     0
     0     0     4
A(:, :, 3) =
     0     0     0
     0     0     6
A(:, :, 4) =
     0     0     0
     0     0     8
>> A(1,2,3)=5 %数组 A 第 3 页的第 1 行第 2 列的元素分别赋值为 5
A(:, :, 1) =
     0     0     0
```

```
0    0    2
A(:,:,2) =
0    0    0
0    0    4
A(:,:,3) =
0    5    0
0    0    6
A(:,:,4) =
0    0    0
0    0    8
```

2. 函数法

在前面章节的叙述中，介绍了 MATLAB 7.0 提供可以生成矩阵及数组的函数，其中部分函数还可以生成高维数组，可以生成高维数组的函数主要包括 zeros、ones、rand、repmat、reshape、cat 等。函数的具体用法在前面章节有详细介绍，这里主要以实例的形式演示通过这些函数如何生成高维数组。

【例 3.48】函数法创建高维数组。

```
>> ones(3,3,3)           %三维全 1 数组的生成
ans(:,:,1) =
1    1    1
1    1    1
1    1    1
ans(:,:,2) =
1    1    1
1    1    1
1    1    1
ans(:,:,3) =
1    1    1
1    1    1
1    1    1
>> zeros(2,2,3)          %三维全零数组的生成
ans(:,:,1) =
0    0
0    0
ans(:,:,2) =
0    0
0    0
ans(:,:,3) =
0    0
0    0
>> rand(2,2,3)           %三维随机阵的生成
ans(:,:,1) =
0.5155    0.4329
0.3340    0.2259
ans(:,:,2) =
0.5798    0.5298
0.7604    0.6405
ans(:,:,3) =
0.2091    0.7833
0.3798    0.6808
>> A=magic(3);
>> B=eye(3);
```

```

>> C=ones(3);
>> cat(3,A,B,C)           %连接 3 个矩阵，每页分别为 1 个矩阵
ans(:,:,1) =
     8     1     6
     3     5     7
     4     9     2
ans(:,:,2) =
     1     0     0
     0     1     0
     0     0     1
ans(:,:,3) =
     1     1     1
     1     1     1
     1     1     1
>> A=eye(2);
>> repmat(A,[2,1,2])      %生成重复数组，在行上复制 2 遍，列上复制 1 遍，页上复制 2 遍
ans(:,:,1) =
     1     0
     0     1
     1     0
     0     1
ans(:,:,2) =
     1     0
     0     1
     1     0
     0     1
>> A=[1:24];
>> reshape(A,2,4,3)       %重置矩阵，生成 2 行 4 列 3 页的高维数组
ans(:,:,1) =
     1     3     5     7
     2     4     6     8
ans(:,:,2) =
     9    11    13    15
    10    12    14    16
ans(:,:,3) =
    17    19    21    23
    18    20    22    24

```

3.8.2 高维数组的基本操作

本小节主要介绍高维数组的基本操作，包括高维数组信息的获取、高维数组元素的访问和高维数组的翻转操作。通过本小节的学习读者将对高维数组的基本操作有全面的了解，达到掌握高维数组基本操作的目的。下面具体讲述这些操作的实现。

1. 高维数组信息的获取

对于高维数组可以像矩阵和数组一样获取其大小信息，判断高维数组的数据类型、是否为空等，这些操作与矩阵操作使用函数都是一样的，下面通过实例演示高维数组基本信息的获取。

【例 3.49】 高维数组信息的获取。

```

>> A=rand(2,3,3);
>> disp(A); %显示高维数组 A
(:,:,1) =

```

```

    0.3200    0.7266    0.7446
    0.9601    0.4120    0.2679
(:, :, 2) =
    0.4399    0.6833    0.8392
    0.9334    0.2126    0.6288
(:, :, 3) =
    0.1338    0.6072    0.3705
    0.2071    0.6299    0.5751
>> isempty(A)           %判断高维数组 A 是否为空
ans =
    0
>> isfloat(A)           %判断高维数组 A 数据类型是否为浮点型
ans =
    1
>> isinteger(A)         %判断高维数组 A 数据类型是否为整型
ans =
    0
>> B=zeros(3,3,3);
>> isequal(A,B)         %判断高维数组 A 和 B 是否相等
ans =
    0
>> size(A)              %获取高维数组 A 行数、列数、页数
ans =
     2     3     3
>> length(A)            %获取高维数组 A 行数、列数、页数中的最大值
ans =
     3
>> numel(A)             %获取高维数组 A 的元素总和
ans =
    18
>> ndims(A)             %获取高维数组 A 的维数
ans =
     3

```

2. 高维数组元素的访问

高维数组元素的访问的方法主要有行、列、页下标法和线性索引法，其中行、列、页下标和线性索引之间的转换可以通过函数 `sub2ind` 和 `ind2sub` 来实现。高维数组元素的访问可以分为单元素、行元素、列元素、页元素和所有元素的共同访问。

【例 3.50】高维数组元素的访问。

```

>> A=rand(2,3,3)        %三维随机数组的生成
A(:, :, 1) =
    0.4514    0.0272    0.0129
    0.0439    0.3127    0.3840
A(:, :, 2) =
    0.6831    0.0353    0.6085
    0.0928    0.6124    0.0158
A(:, :, 3) =
    0.0164    0.5869    0.3676
    0.1901    0.0576    0.6315
>> A(:, 2, :)           %访问三维数组 A 中所有行、所有页的第 2 列元素
ans(:, :, 1) =

```

```

    0.0272
    0.3127
ans(:,:,2) =
    0.0353
    0.6124
ans(:,:,3) =
    0.5869
    0.0576
>> A(:,2,1) %访问三维数组 A 中第 1 页第 2 列, 所有行的元素
ans =
    0.0272
    0.3127
>> A(1,2,3) %访问三维数组 A 中第 1 行、第 2 列、第 3 页的元素
ans =
    0.5869
>> A(1,:,3) %访问三维数组 A 中第 1 行第 3 页的所有列的元素
ans =
    0.0164    0.5869    0.3676
>> A(:) %访问三维数组 A 中的所有元素, 并按线性索引输出
ans =
    0.4514
    0.0439
    0.0272
    0.3127
    0.0129
    0.3840
    0.6831
    0.0928
    0.0353
    0.6124
    0.6085
    0.0158
    0.0164
    0.1901
    0.5869
    0.0576
    0.3676
    0.6315
>> sub2ind(size(A),1,2,3) %高维数组的行列页下标转换为线性索引坐标
ans =
    15
>> [i,j,k]=ind2sub(size(A),15) %高维数组的线性索引坐标转换为行列页下标
i =
    1
j =
    2
k =
    3

```

3. 高维数组的翻转操作

MATLAB 7.0 中提供了函数 `flipdim` 用于实现高维数组的翻转操作, 函数的调用格式如下。

```
B = flipdim(A,dim)
```

其中, A 为需要翻转的数组, dim 为翻转的基准维数, 当 dim 为 3 时即对高维数组按页翻转。

【例 3.51】高维数组的翻转操作。

```
>> A=rand(2,2,3)
A(:,:,1) =
    0.8656    0.8049
    0.2324    0.9084
A(:,:,2) =
    0.2319    0.0498
    0.2393    0.0784
A(:,:,3) =
    0.6408    0.8439
    0.1909    0.1739
>> B = flipdim(A,3)           %高维数组按页翻转
B(:,:,1) =
    0.6408    0.8439
    0.1909    0.1739
B(:,:,2) =
    0.2319    0.0498
    0.2393    0.0784
B(:,:,3) =
    0.8656    0.8049
    0.2324    0.9084
```

3.9 本章小结

本章主要介绍了 MATLAB 矩阵和数组的常用操作。矩阵和数组是 MATLAB 数据分析操作的主要对象, 因此学好本章的基础知识, 将有利于读者尽快掌握 MATLAB 的常用操作, 编写简单的代码。

通过本章的学习读者应该掌握:

- 矩阵和数组的概念。
- 矩阵和数组的创建方法。
- 矩阵和数组的基本操作函数。
- 矩阵和数组的简单运算。
- 矩阵和数组的逻辑运算。
- 向量的基本知识。
- 高维数组的操作。


第4章


程序设计

在前面的章节中我们对 MATLAB 基础知识做了详尽的介绍，主要是在 MATLAB 命令窗口演示这些基本函数、功能。而 MATLAB 作为一门优秀的编程语言也同其他高级语言一样，可以进行复杂程序的设计。本章我们将学习 MATLAB 程序设计的相关知识，主要涉及程序设计的文件类型、变量和常量、流程控制、调试与优化等。在本章的学习过程中我们会发现 MATLAB 和其他编程语言程序设计的相同或相近之处，但是希望读者在发现 MATLAB 和其他编程语言的共性之处的同时能够深刻体会不同之处。勤于思考，不要停留在原来的编程习惯中，换一种编程方式，可能取得更好的效果。另外，在程序出错后，对程序的调试、修改、完善，也是用好 MATLAB 解决实际问题的关键。




4.1 程序设计概述

MATLAB 的程序设计是在 M 文件中进行的。M 文件按照调用方式的不同可以分为脚本文件和函数文件。程序设计中涉及的主要步骤如下。

(1) 程序文件的新建：通过单击主窗口工具条上的  图标，新建如图 4.1 所示的 M 文件，在 M 文件内即可以开始编写程序。

(2) 程序文件的保存：程序编写完毕后，一般要先保存才可运行程序。程序的保存可以通过工具条上的  图标或者按“Ctrl+S”组合键，打开如图 4.2 所示的程序保存窗口，给 M 文件命名，然后保存。

MATLAB 中 M 文件的命名规则如下。

- M 文件名需以字母为首字母，文件名中其他字符可以包括英文字母、数字和下划线。
- M 文件名不能与 MATLAB 内部的函数重名。在实际应用中要严格遵守 MATLAB 中 M 文件的命令规则，否则会产生很多不可预料的错误，并且此时程序本身并没有错，而运行结果出错，会给程序的调试工作带来很多问题。
- 程序的运行：程序编写完单击工具条上的图标  或者按“F5”键即可运行程序。
- 程序的调试：如果运行的程序出错，可以通过菜单中的 Debug 菜单项和工具栏中  调试工具进行调试。
- 程序的布局显示：图标  用于显示 M 文件窗口的布局，可以同时显示多个 M 文件，方便文件的比对。

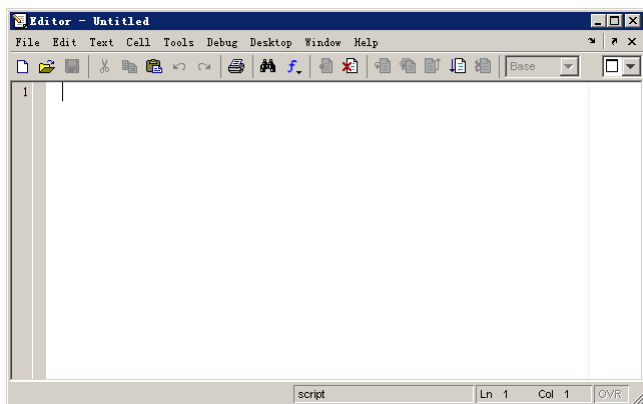


图 4.1 M 文件窗口

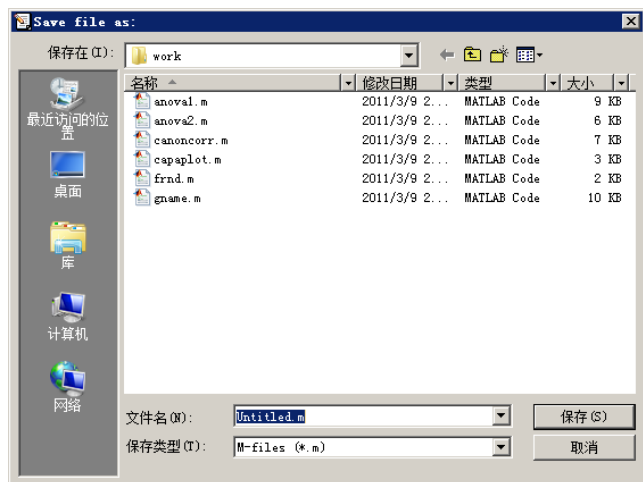


图 4.2 M 文件的保存

M 文件窗口的菜单项包括“File”、“Edit”、“Text”、“Cell”、“Tools”、“Debug”、“Desktop”、“Window”、“Help”。其中，“File”、“Edit”、“Desktop”、“Window”、“Help”与主界面中相应菜单的功能基本一致，具体参见本书第1章的介绍。而程序调试菜单项“Debug”的介绍将在4.7节中详细叙述，“Tools”菜单将在代码优化内容中介绍，这里主要介绍“Text”、“Cell”两个菜单项的使用。

1. “Text”菜单项

“Text”菜单项主要用于对 Editor 窗口编写的代码进行格式上的操作，包括代码缩进的控制、执行、注释的控制。如图 4.3 所示，下面具体介绍各菜单项的使用。

- Evaluate Selection: 运行所选的代码。
- Wrap Selected Comments: 隐藏所选的代码。
- Comment: 选择的内容设置为注释。
- Uncomment: 取消选择的内容作为注释。
- Decrease Indent: 减少所选代码的缩进。
- Increase Indent: 增加所选代码的缩进。
- Smart Indent: 按照编程规范智能缩进代码。

Evaluate Selection	F9
Wrap Selected Comments	
Comment	Ctrl+R
Uncomment	Ctrl+T
Decrease Indent	Ctrl+[
Increase Indent	Ctrl+]
Smart Indent	Ctrl+I

图 4.3 Editor 窗口的“Text”菜单项

2. “Cell”菜单项

“Cell”菜单项主要用于控制 Cell 模式的使用，如图 4.4 所示，Cell 模式是 MATLAB 的一大特色，通过 Cell 模式的使用可以让用户在调试某一段代码块的时候，方便地重复运行，同时也向用户提供了代码块的思想。不同的功能可以设计成不同的代码块，分块执行代码，便于程序的执行、查错。

一段代码以%%标记后加一个空格即为代码块生成了 Cell 模式，光标移动到 Cell 模式中时，背景将变为浅黄色。按“Ctrl+Enter”组合键或者选择“Cell”→“Evaluate Current Cell”命令控制执行当前 Cell 模块下的代码。同时 Cell 模式的开启与关闭分别通过菜单项 Enable Cell Mode 和 Disable Cell Mode 控制。

Disable Cell Mode	
Evaluate Current Cell	Ctrl+Enter
Evaluate Current Cell and Advance	Ctrl+Shift+Enter
Evaluate Entire File	
Insert Cell Divider	
Insert Cell Dividers around Selection	
Insert Text Markup	
Next Cell	Ctrl+Down
Previous Cell	Ctrl+Up

Cell Title
Descriptive Text
Bold Text
Monospaced Text
Preformatted Text
Bulleted List
TeX Equation

图 4.4 Editor 窗口的“Cell”菜单项

同时，MATLAB 上述菜单项的功能也可以通过 M 文件编辑窗口右键快捷菜单打开，如图 4.5 所示。包括代码执行、注释的操作、程序调试设置等，用法同各菜单中的菜单项，在此不再详细展开叙述。

Evaluate Selection	F9
Open Selection	Ctrl+D
Help on Selection	
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Wrap Selected Comments	
Comment	Ctrl+R
Uncomment	Ctrl+T
Smart Indent	Ctrl+I
Evaluate Current Cell	Ctrl+Enter
Insert Cell Divider	
Insert Cell Dividers around Selection	
Insert Text Markup	
Set/Clear Breakpoint	F12
Set/Modify Conditional Breakpoint...	
Enable/Disable Breakpoint	
Go Until Cursor	

图 4.5 Editor 窗口的右键快捷菜单

4.2 脚本文件

脚本文件是由多条命令组成的文件。对于需要在命令行窗口输入多行命令才能完成的任务，可以考虑在 MATLAB 代码编写窗口建立脚本文件，便于操作，而且文件可以保存，可以重复执行。

简单来讲，脚本文件就是由一条条命令组成的，可以直接访问 MATLAB 当前工作空间内的变量，同时脚本文件内代码生成的变量也将保存在 MATLAB 的工作空间内。

按“F5”键或者单击标题栏中的运行图标，即可运行脚本文件，MATLAB 程序执行机制是会对脚本文件先保存，然后再执行。或者可以选中需要运行的代码，然后选择“Evaluate Selection”命令，执行选中的代码。

通过“F5”键或者运行图标完整执行脚本文件时，脚本文件需要在 MATLAB 的工作路径下，如果不在，运行后会弹出如图 4.6 所示的弹出式窗口。为了运行该脚本文件，用户需要把脚本文件所在的路径设置为当前路径设置，或者添加脚本文件所在的路径到 MATLAB 的搜索路径。而通过“Evaluate Selection”命令执行脚本文件则不需要设置路径，因为此时类似于在命令窗口运行多行代码。

脚本文件的调用直接输入函数名即可，此时脚本文件同样需要在 MATLAB 的搜索路径中。

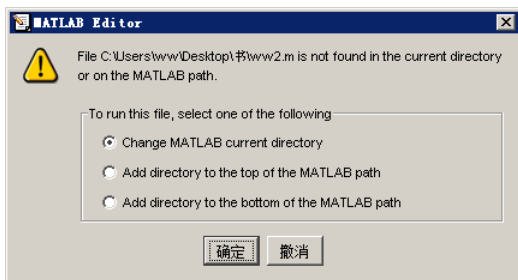


图 4.6 Editor 窗口设置脚本文件路径的弹出式窗口

4.3 函数文件

需要通过调用才可以执行的一类 M 文件为函数文件，函数文件一般有输入、输出参数，可以供不同的输入参数重复调用产生不同的输出结果。函数文件中产生的变量只在函数运行过程中有效，当函数执行完后，变量即不存在。函数文件一般用于需要让不同的参数重复调用的代码，但是函数文件的执行类似于“黑箱”，我们无法看到函数文件执行的中间过程，因此会给程序的调试带来比较大的麻烦，对于入门读者建议最初以编写脚本文件为基础，当编写的脚本文件运行后完整无误，可以考虑再设计成函数。

目前 MATLAB 软件提供了大量的现场函数供我们直接调用，大大提高了代码的编写效率，而一些新开发的算法网上也经常能找到现成的函数文件供我们使用。

4.3.1 函数的定义

当 M 文件的第一行为 function 的函数定义语句，即函数文件。完整的 MATLAB 函数文件由函数定义行、H1 行、帮助文本、函数体、注释组成。例 4.1 即 MATLAB 自带的函数 mean() 一个完整的函数文件。

【例 4.1】完整的函数文件。

在 MATLAB 命令窗口输入 `edit mean`，即可打开 MATLAB 自带的函数 `mean()` 的 M 文件，如下所示。主要包含函数的定义行、H1 行、帮助文本和函数体，后面的内容会具体介绍这几部分。

```
function y = mean(x,dim)
%MEAN Average or mean value.
% For vectors, MEAN(X) is the mean value of the elements in X. For
% matrices, MEAN(X) is a row vector containing the mean value of
% each column. For N-D arrays, MEAN(X) is the mean value of the
% elements along the first non-singleton dimension of X.
%
% MEAN(X,DIM) takes the mean along the dimension DIM of X.
%
% Example: If X = [0 1 2
%                  3 4 5]
%
% then mean(X,1) is [1.5 2.5 3.5] and mean(X,2) is [1
%                                                    4]
%
% Class support for input X:
%   float: double, single
%
% See also MEDIAN, STD, MIN, MAX, COV.

% Copyright 1984-2004 The MathWorks, Inc.
% $Revision: 5.17.4.1 $ $Date: 2004/03/09 16:16:26 $

if nargin==1,
    % Determine which dimension SUM will use
    dim = min(find(size(x)~=1));
    if isempty(dim), dim = 1; end
    y = sum(x)/size(x,dim);
else
    y = sum(x,dim)/size(x,dim);
end
```

1. 函数的定义行

函数的定义行位于 M 文件的第一行，可以作为与脚本文件的区别之一。函数定义行的格式如下。

- `function [y1,y2,...]=fun_name(x1,x2,...)`: 多输入参数，多输出参数。
- `function y1=fun_name(x1,x2,...)`: 多输入参数，单输出参数。
- `function =fun_name(x1,x2,...)`: 多输入参数，无输出参数。

同时输入参数也可以为单输入参数或者无输入参数。

其中，“function”为 MATLAB 语言中函数的标识的关键字符，`fun_name` 为函数名。`x1`、`x2` 为函数输入变量，输入变量是由小括号标识的，各变量间用逗号间隔；而 `y1`、`y2` 为输出变量，由中括号标识，各变量间用逗号间隔。同时需要注意到函数的输入变量引用的只是变量的值，函数体中对变量操作所引起变量值的改变不会保存到 MATLAB 工作空间内。

例 4.1 中语句“`function y = mean(x,dim)`”即函数的定义行，此函数包括两个输入参数，返回一个输出参数 `y`。

2. H1 行

H1 行为紧接函数定义行后的第一行以百分号“%”开头的语句。此语句在通过函数 `lookfor` 查询 MATLAB 函数的简单帮助文档时会显示。例 4.1 中的 H1 行语句为“%MEAN Average or mean value.”。

3. 帮助文本

在命令窗口中通过 help 命令获取的函数帮助信息即为此部分的帮助文本，一般包括函数的功能，可以调用的不同格式、参数，一些简单的调用实例等。帮助文本在函数文件中为 H1 行后的连续的以“%”开头的文本。

例 4.1 中函数 mean 的帮助文本为从语句“% For vectors, MEAN(X) is the mean value of the elements in X. For”开始到“% \$Revision: 5.17.4.1 \$ \$Date: 2004/03/09 16:16:26 \$”语句结束的连续的注释文档。

4. 函数体

函数体是函数文件的核心，包括了实现函数功能的所有命令的集合。函数中可以有赋值语句、程序流程控制语句、运算语句、其他函数的调用语句等。

【例 4.2】函数文件的函数体如下。

```
if nargin==1,
    % Determine which dimension SUM will use
    dim = min(find(size(x)~=1));
    if isempty(dim), dim = 1; end
    y = sum(x)/size(x,dim);
else
    y = sum(x,dim)/size(x,dim);
end
```

5. 注释

注释是以符号“%”起始对程序代码的解释文字。MATLAB 程序执行时不运行“%”后的语句。如果在一行语句的初始为“%”号，该行语句即注释语句，程序运行时不执行，这样在程序调试的时候是很有帮助的，对于一些暂时不希望执行的语句，可以添加注释号。注释号的添加可以通过手动在注释代码前输入百分号或者通过菜单和快捷方式中的命令“Comments”。

阅读程序的注释便于读懂程序，建议读者在编写代码时尽量养成多写注释的习惯，这样可以方便别人阅读，同时也利于自己修改。

由上述 5 个部分可以构成一个完整的函数，但是用户在实际编写函数中，这 5 个部分除了函数定义行与函数体是必须的，其他部分可写可不写，但是对于适量的注释行笔者建议还是很有必要的。

4.3.2 函数类型

MATLAB 的函数类型主要有主函数、子函数、私有函数、嵌套函数、重载函数、匿名函数。这里主要介绍常用的主函数、子函数和匿名函数。

1. 主函数和子函数

出现在函数文件最前面的函数即为主函数，之后的函数为子函数。一个函数文件可以包括多个子函数，但只能有一个主函数。函数文件名同主函数名。函数文件中的各子函数是并列的关系，没有顺序关系，只有调用的前后关系，且只能被该函数文件的主函数和子函数调用。

【例 4.3】主函数和子函数。

```
function y=compute(x)
y=fun1(x)+fun2(x);
    function z=fun1(x)
        z=x*10-x^2;
    end
    function z=fun2(x)
        z=x^3;
    end
end
```

其中，函数 `compute()` 为主函数，`fun1` 和 `fun2` 为子函数，在主函数中调用了子函数。

2. 匿名函数

匿名函数是最简单的函数形式，不需要函数名、M 文件，仅需要通过函数句柄的使用即可方便地调用匿名函数，其调用格式为“`f=@(变量列表)函数表达式`”。

函数句柄可以让函数赋值给变量，其调用格式为变量名=`@`函数名，此处函数名可以为任意函数名，赋值后即可像使用变量一样使用函数，通过“变量名”可以方便地调用原函数。函数句柄的使用可以提高代码的运行速度，同时便于不同目录下函数的调用。

【例 4.4】匿名函数的使用。

```
>> f=@(x,y)x+y;           %定义匿名函数，等价于原函数 z=fun1(x,y)  z=x+y;
>> f(2,3)
ans =
    5
```

4.3.3 函数的调用和变量传递

函数调用的一般格式为：`[y1,y2,...]=函数名(x1,x2,...)`

其中，`x1`、`x2` 为函数的输入参数，传递的是参数的值；而 `y1`、`y2` 为函数的输出参数，获取的是函数调用后的值，为实参，将存储到 MATLAB 工作空间中。

在函数调用过程中 MATLAB 查询的是文件名，而调用格式中又为函数名，因而为避免不必要的麻烦，存储函数文件时文件名应当与主函数名相一致，以便于理解和使用。

函数调用的注意事项如下：

- 函数调用时输入、输出参数的顺序应该与函数定义的相同，输出参数个数可少于函数中定义的输出参数个数，输入参数个数：如果函数文件中给出输入参数的默认值，则可以少于函数定义的输入参数的个数。但是输入、输出参数个数不可多于函数定义的输入、输出参数个数。
- 在调用函数时，MATLAB 的调用机制是搜索当前工作空间内是否有此函数名的变量，因此调用函数前需要检查一下工作空间内是否有与待调用的函数同名的变量。
- 函数调用后的输出参数将保存到 MATLAB 存储空间。如果有多个输出参数，但是在函数调用时未列出，则在工作空间内不会存有未调用的输出参数。

【例 4.5】函数的调用。

```
%定义函数
function y=fun_chuandi(x)
x=x^2;
y=2*x;
end
```

保存上述函数文件，文件名为“`fun_chuandi`”。在命令窗口中调用此函数，并观察参数的传递。

```
>> x=2;
>> y=fun_chuandi(x) %调用函数 fun_chuandi(), 调用后获得函数实参 y, 而输入参数 x 值不变
y =
    8
>> x
x =
    2
>> x2=5;
>> y=fun_chuandi(x2) %函数传递的仅是输入参数的值, 输入参数为新的变量 x2 时同样获得返回参数
y =
   50
>> y=fun_chuandi(4) %函数的输入参数可以直接为某一具体的值
y =
   32
```

4.3.4 输入/输出参数的控制

MATLAB 提供了函数 `nargin` 和 `nargout` 控制函数输入/输出参数，其调用格式如下。

- `n = nargin`: 用于获取当前函数文件内的输入参数个数。
- `n = nargin('fun')`: 用于获取指定函数文件 `fun` 内的输入参数个数。
- `n = nargout`: 用于获取当前函数文件内的输出参数个数。
- `n = nargout('fun')`: 用于获取指定函数文件 `fun` 内的输出参数个数。

一般函数中输入/输出参数的检测主要用于判断用户在调用函数时是否给出了充分的调用参数或者根据调用参数的个数执行不同的函数功能，用户没有提供的输入参数可以设置为函数自带的默认值。

【例 4.6】输入和输出参数的控制。

```
function [y1 y2 y3]=funtest(x1,x2,alpha)
%输入参数个数控制
if nargin < 2                %如果输入参数个数小于函数必须的两个，程序提示出错
    error('FUNCTION requires at least two input arguments.');
```

elseif nargin == 2 %如果输入参数个数等于函数必须的两个，第3个参数使用函数提供的默认值

```
    alpha = 0.05;
end
y1=alpha*(x1+x2);
if nargout >= 2              %如果输出参数的个数大于等于两个，计算第2个输出参数值
    y2=y1.^2;
    if nargout >= 3          %如果输出参数的个数大于等于3个，计算第3个输出参数的值
        y3=2*y2;
    end
end
end
```

不同的输入、输出参数调用此函数：

```
>> y1=funtest(1,2,0.3)      %只计算函数一个输出参数
y1 =
    0.9000
>> funtest(1,2,0.3)         %如果不给出函数输出参数，默认状态下获得第1个输出参数的值
ans =
    0.9000
>> y1=funtest(1,2)          %调用函数时只给出函数两个输入参数，第3个参数使用函数自带的默认值
y1 =
    0.1500
>> [y1,y2]=funtest(1,2)     %调用函数，输出参数为两个
y1 =
    0.1500
y2 =
    0.0225
>> [y1,y2,y3]=funtest(1,3)  %调用函数，输出参数为3个
y1 =
    0.2000
y2 =
    0.0400
y3 =
    0.0800
>> [y1,y2]=funtest(1)       %调用函数，一个输入参数，小于函数必须的两个输入参数，函数给出警告
??? Error using ==> funtest
FUNCTION requires at least two input arguments.

>> [y1,y2,y3]=funtest(1,3,1,1) %调用函数，函数的输入参数过多，提示出错
??? Error using ==> funtest
Too many input arguments.
```



```
>> [y1,y2,y3,y4]=funtest(1,3)    %调用函数，函数的输出参数过多，提示出错
??? Error using ==> funtest
Too many output arguments.
```

4.4 常量、变量

变量是程序设计语言的重要组成元素。MATLAB 语言中的变量与常规程序设计语言不同的是不需要事先对所使用的变量进行声明，也不需要指定变量的数据类型。MATLAB 语言会自动根据变量的赋值情况产生默认的数据类型。MATLAB 中也提供了一些值不变的变量，为常量。这些常量一般具有固定的含义。

本节中将重点介绍变量的命令、变量类型和常量的相关知识。

4.4.1 变量的命名

MATLAB 中的变量在使用前不需要声明，指定数据类型，但是变量的命令需要遵守如下规定：

- 变量名以字母开头，且只能由字母、下画线和数字混合组成。
- 变量名区分大小写。
- 变量名的长度不要超过 63 个。
- 变量名尽量不要与已有的函数名、常量名相同。
- 变量名不可以使用 MATLAB 中的关键字，MATLAB 中的关键字包括“break”、“case”、“catch”、“continue”、“else”、“elseif”、“end”、“for”、“function”、“global”、“if”、“otherwise”、“persistent”、“return”、“switch”、“try”、“while”。

4.4.2 系统预定义的常量

MATLAB 语言包含一些预定义的变量，这些特殊的变量称为常量。如表 4.1 所示列出了 MATLAB 7.0 中的主要常量。

表 4.1 MATLAB 7.0 中的主要常量

常 量	含 义
pi	圆周率
eps	浮点运算的精度，为 2.2204e-016
inf	正无穷大
NaN	表示不定值
realmax	最大的浮点数
realmin	最小的浮点数
i,j	虚数单位

【例 4.7】系统预定义的常量的使用。

```
>> s=2^2*pi          %计算半径为 2 的圆的面积
s =
    12.5664
>> pi               %查看常量 pi 的值
ans =
    3.1416
>> pi=3.14          %改变常量 pi 的值，类似于定义了一个新的常量 pi
pi =
    3.1400
```

```
>> s=2^2*pi
s =
    12.5600
>> clear pi           %清除改变常量值 pi 后，重新查看常量的值仍为原来的值
>> pi
ans =
    3.1416
```

4.4.3 变量类型

MATLAB 中的变量可以分为局部变量与全局变量，其作用域不同。默认状况下的变量都为局部变量，只能在当前的工作空间或者函数体中有效。函数文件中的变量都是局部的，即一个函数文件中定义的变量不能被另一个函数文件使用，也不存储到 MATLAB 工作空间内。如果不通过函数返回，则在函数运行完毕后局部变量不存在。

如果一些变量需要在不同的函数体、工作空间中同时使用，需要定义为全局变量。全局变量定义的格式如下。

`global` 变量名列表

其中，不同变量间用空格隔开。全局变量的定义需要在函数体最开始的位置，保证全局变量在其他函数中的使用，而其他的函数或工作空间内要使用这个全局变量时，需要重新声明。全局变量的定义只要在一般的变量前加关键字 `global` 即可。全局变量在使用时要特别注意，任意一处使用全局变量导致其值改变，全局变量的值就永久地改变了，对于大型程序的调试是很不方便的。

4.4.4 系统预定义的变量

MATLAB 中也有一些固定的变量，有特殊的含义，具体如表 4.2 所示。

表 4.2 MATLAB 7.0 中的系统预定义的变量

变 量	含 义
ans	默认状态下的赋值变量
nargin	函数输入参数个数
nargout	函数输出参数个数
varargin	可变的函数输入参数个数
varargout	可变的函数输出参数个数
lasterr	try...catch 语句中保存出错信息

【例 4.8】系统预定义的变量的使用。

(1) ans 的使用。

```
>> 2*2           %如果计算的表达式计算后没有赋值给一个变量，默认状态下保存到变量 ans 中
ans =
    4
```

(2) varargin 和 varargout 的使用。

```
function varargout=varartest(varargin)
a=mean(cell2mat(varargin));           %可变的输入变量 varargin 数据类型为 cell 型，转换为数值型
for i=1:nargout                       %nargout 内存储函数输出参数个数，为每一个输出参数赋值
    varargout{i}=i.^2-a;               %可变的输出参数 varargin 数据类型应为 cell 型
end
end
```

在命令窗口中调用如下函数。

```
>> [y1 y2 y3]=varartest(1,3,5)
y1 =
   -2
y2 =
```

$$y^3 = \frac{1}{6}$$

4.5 程序结构及流程控制

MATLAB 与一般程序设计的高级语言类似,程序结构可以分为顺序结构、循环结构和条件结构。顺序结构是按照代码书写的先后顺序执行,是最常用的结构,一般的程序中都会有顺序结构的成分。循环结构用于重复地执行某一段代码,通过循环流程控制语句控制程序的执行。条件结构可用于在一定条件下执行相应的代码,通过条件流程控制语句执行代码。

MATLAB 提供了 8 种常用的控制程序流程语句,分别为“for”、“while”、“if”、“switch”、“try”、“continue”、“break”、“return”语句。流程控制语句一般比较长,需要连续地书写一段代码,因此一般流程控制语句多在函数文件或者脚本文件中使用。本节将在不同程序结构的代码中详细介绍流程控制语句的使用。通过本节的学习,将使读者掌握 MATLAB 基础程序设计流程的基础知识,指导读者写出简洁的程序。

4.5.1 赋值语句

MATLAB 中典型的赋值语句调用格式如下。

变量名=表达式

其中,表达式可以是运算表达式,也可以是函数调用等。表达式后加“;”,赋值表达式的变量值不显示;如果不加“;”,将在命令窗口中显示赋值表达式。一般对于数据量比较大的变量建议加上“;”,大数据量变量的显示会减慢程序执行的效率。而如果希望直接在命令窗口看到函数的数据显示,则可以不加“;”,特别是在函数中,因为函数在运行完后中间执行的变量不会保存下来,我们可以在希望看到中间变量的值后不加“;”,观察命令窗口的显示。

【例 4.9】赋值语句的使用实例。

(1) 矩阵的赋值。

```
>> A=[1 2 3;4 5 6]
A =
     1     2     3
     4     5     6
```

(2) 表达式的运算。

```
>> y=2*log(10)+2^3
y =
    12.6052
```

(3) 函数的调用。

```
>> a=length(A)
a =
     3
```

4.5.2 条件语句

MATLAB 中提供了条件语句,用于控制在一定的条件下执行特定的代码。常用的条件语句有 if 语句、switch 语句和 try 语句。

1. if 语句

在 MATLAB 7.0 中,if 语句有 3 种使用形式。

(1) 单分支形式: if...end。

语句格式如下。

```
if 条件表达式
语句组
end
```

当条件表达式为“真”时，则执行语句组；当条件表达式为“假”时，则直接执行 if 语句后面的语句。

(2) 双分支形式: if...else...end。

语句格式如下。

```
if 条件表达式
语句组 1
else
语句组 2
end
```

当条件表达式为“真”时，执行语句组 1 的代码；当条件表达式为“假”时，执行语句组 2 的代码，执行完语句组 1 或语句组 2 的代码后，再执行 if 语句后的语句。

(3) 多分支形式: if...elseif... else...end。

语句格式如下。

```
if 条件表达式 1
语句组 1
elseif 条件表达式 2
语句组 2
.....
elseif 条件表达式 m
语句组 m
else
语句组 m+1
end
```

当条件表达式 1 为“真”时，执行语句组 1 的代码，否则转入判断条件表达式 2，若条件表达式 2 为“真”，则执行语句组 2 的代码，否则转入判断条件表达式 3，依此类推。若各 elseif 语句的条件表达式都不为“真”，则转入 else 语句，执行语句组 m+1 的代码。

【例 4.10】 if 语句的使用。

```
x=[5 2 3;4 2 4;5 6 8];
%单分支的 if 语句
for i=1:3
    for j=1:3
        if x(i,j)>5
            disp(i)
            disp(j)
        end
    end
end
```

执行单分支 if 语句，命令窗口显示运行结果如下。

```
3
2
3
3
%双分支的 if 语句
n=0;
for i=1:3
    for j=1:3
        if x(i,j)>5
            disp(i)
            disp(j)
        else
            n=n+1;
        end
    end
end
```

```

        end
    end
end

```

执行双分支 if 语句，命令窗口显示运行结果如下。

```

    3
    2
    3
    3
%多分支语句
x=5;
if x<0
    disp('x 小于 0');
elseif x==0
    disp('x 等于 0');
else
    disp('x 大于 0');
end

```

执行多分支 if 语句，命令窗口显示运行结果如下。

x 大于 0

2. switch 语句

多分支语句的条件判断也可以使用函数 switch，与 if 多分支语句相比，switch 在判断变量单一，而判断条件多时使用比较方便。

switch-case 语句的一般形式如下。

```

switch 表达式
case 表达式 1
    语句组 1
case 表达式 2
    语句组 2
.....
case 表达式 m
    语句组 m
otherwise
    语句组 m+1
end

```

用于判断的表达式必须为一标量或者字符串，依次判断其是否等于表达式 1、2... 的值，如果符合则执行相应的语句组，且程序将跳出 switch-case 语句，不再对后面的语句进行判断。如果和所有的 case 语句后的表达式都不符合，则直接运行 otherwise 语句后的语句组的代码。

【例 4.11】 switch 语句的使用。

1) switch 语句用于标量值的多分支判断

在代码编辑窗口输入如下的代码，用于根据输入的成绩分数判断成绩优秀、良好、合格或者不合格，保存脚本文件，函数名为 switchtest。

```

grade=input('请输入考试成绩');
switch round(grade/10)
    case {9,10}
        disp('成绩优秀');
    case {7,8}
        disp('成绩良好');
    case {6}
        disp('成绩合格');
    otherwise
        disp('成绩不合格');
end

```

在命令窗口调用函数，测试函数 switchtest 的使用。

```
>> switchtest
请输入考试成绩 90           %输入成绩值“90”，返回“成绩优秀”
成绩优秀
>> switchtest
请输入考试成绩 80           %输入成绩值“80”，返回“成绩良好”
成绩良好
```

2) switch 语句用于字符串的多分支判断

在代码编辑窗口输入如下的代码，用于根据输入的城市名返回相应的区号，保存为脚本文件，函数名为 switchtest。

```
%输入城市名给出相应的区号
country=input('请输入城市名','s');           %通过交互式语言，用于输入城市名
country=lower(country);                       %输入的城市名的拼音转换为小写字母
switch country                                %country 的多分支判断
    case 'nanjing'
        disp('025');
    case 'beijing'
        disp('010');
    case 'shanghai'
        disp('021');
    case 'guangzhou'
        disp('020');
    case 'tianjin'
        disp('012');
    case 'wuhan'
        disp('027');
    otherwise
        disp('不在查询范围');
end
```

在命令窗口调用函数，测试函数 switchtest2 的使用。

```
>> switchtest2
请输入城市名 nanjing
025
>> switchtest2
请输入城市名 Shanghai
021
```

3. try 语句

try 语句主要用于对程序出错的检测，并在出错后采取相应的措施。

try 语句的一般形式如下。

```
try
语句组 1
catch
语句组 2
end
```

try 首先执行语句组 1 的代码，如果出错，则执行语句组 2 的代码。try 语句用于提高代码的容错能力。如果需要查看程序的出错信息，可以查看系统的预定义变量 lasterr，lasterr 变量可用于显示函数的出错信息。

【例 4.12】try 语句的使用。

```
%矩阵正常的加法运算
x=rand(2);
y=eye(2);
try
    z=x+y;
    disp(z)
catch
```

```

    errordlg('矩阵相加有误','error')
end

%矩阵不正常的加法运算
x=rand(2);
y=eye(3);
try
    z=x+y;
    disp(z)
catch
    errordlg('矩阵相加有误','error')
end

```

运行程序，出错后显示如图 4.7 所示的出错提示框，同时在命令窗口查看出错信息。

```

>> lasterr
ans =
Error using ==> plus
Matrix dimensions must agree.

```

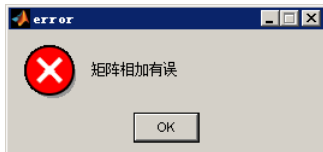


图 4.7 程序出错提示框

4.5.3 循环语句

循环语句一般用于在一定的条件下需要重复执行的程序。MATLAB 中提供的循环控制语句有 for 语句、while 语句、continue 语句等。

1. for 语句

使用 for 循环语句可以按照指定的次数重复执行循环体内的语句。

for 循环语句的调用形式如下。

```

for 循环变量=表达式 1:表达式 2:表达式 3
    循环体语句
end

```

for 循环语句执行时首先计算三个表达式的值，表达式 1 为循环条件控制的初始变量，最初循环变量的值为循环表达式 1 的值，表达式 3 为循环终值。此时判断循环变量是否在表达式 1 和表达式 3 之间，如果在则执行循环体。表达式 2 为循环步长，在执行完循环体后，循环变量增加一个表达式 2 的值，如果此时循环变量的值介于表达式 1 和表达式 3 的值之间，则执行循环体语句，否则结束循环的执行。

当循环变量不满足循环条件即结束 for 语句的执行，而继续执行 for 语句后的语句。循环步长可以为默认，默认状态下循环步长为 1；当循环步长为正数时，要求循环初值小于终值，而当步长为负数时，要求循环初值大于终值，否则循环语句不执行。当循环步长可以整除循环终值与循环初值的差值时，循环次数 = (循环终值 - 循环初值) / 步长。

for 循环的使用注意事项：在 for 循环中不可以出现对循环变量的重新赋值操作，否则循环将出错。for 循环可以嵌套使用。

【例 4.13】for 循环语句的使用。

```

%for 循环程序演示
for i=2:2:6 %初值为 2，步长为 2，终值为 6 的循环，循环体共被执行 3 次
    disp(i);
end

```

命令窗口的显示结果如下。

```

2
4
6
%使用默认状态下的循环步长
for i=2:6 %默认状态下循环步长为 1
    disp(i);
end

```

命令窗口的显示结果如下。

```

2
3
4
5
6
%循环步长为负数
for i=6:-2:2 %循环步长为-2
    disp(i);
end

```

命令窗口的显示结果如下。

```

6
4
2
%不可执行的循环
for i=1:-2:6 %循环步长为负数，而循环初值小于循环终值时，循环不执行
    disp(i);
end

```

循环不执行，命令窗口无结果显示。

```

%嵌套循环的使用
for i=1:3
    for j=1:3
        a(i,j)=i+j;
    end
end
disp(a)

```

通过循环嵌套生成矩阵 a。

```

2    3    4
3    4    5
4    5    6

```

2. while 语句

当事先无法准确确定循环次数时，可以使用 while 语句，其一般的格式如下。

```

while 表达式
    循环体
end

```

其中，表达式为循环的判断条件，一般为逻辑表达式。当循环条件为“真”，则执行循环体，直至循环条件为“假”，终止循环。如果循环判断表达式为矩阵时，当且仅当矩阵内的所有元素都为非零时，逻辑表达式的值为“真”。

在 while 语句循环体中一般都有修改循环控制表达式值的命令，使循环条件由真到假，否则循环将陷入死循环中。但是流程控制语句中的 break 语句可以使程序终止，不用满足循环条件为假的条件，退出循环。

while 语句与 for 语句在循环控制中最大的不同之处主要在于 while 语句只要循环条件为真，即循环要一直进行下去，而 for 循环，循环是有一定的次数的，超过一定次数后，循环即终止。读者在设计程序时需要根据实际的需要选择不同的循环控制语句。

【例 4.14】while 循环语句的使用。

```
%计算 100+99+...+1
s=100;
n=0;           %用于获取循环次数
sum=0;         %用于累加求和
while s>0      %循环执行条件为变量 s 大于 0
    sum=sum+s;
    s=s-1;     %循环判断条件改变
    n=n+1;     %每进行一次循环, 循环次数 n 加 1
end
```

在 MATLAB 代码编辑器窗口输入上述代码, 保存为脚本文件, 运行代码。在命令窗口查看程序运行后变量值的变化。

```
>> s           %循环终止时变量 s 的值为 0, 不满足循环条件为真
s =
    0
>> sum         %求和的结果为 5050
sum =
    5050
>> n           %循环次数为 100
n =
    100
```

3. continue 语句

MATLAB 中提供了流程控制语句 continue 命令用于控制循环, 当程序运行到该命令时会忽略其后的循环体语句, 而转入下一次的循环。continue 语句的调用格式为: continue。

【例 4.15】continue 语句的使用。

```
%计算向量 a 的倒数
a=[1 3 0 2 3 4 0 5 7 5];
for i=1:10
    if a(i)==0           %当向量 a 中的元素为 0 时, 跳出当前循环, 进入下一次循环
        b(i)=0;
        continue;
    end
    b(i)=1./a(i);
end
```

4. break 语句

当程序流程运行至 break 命令时, 则不论循环判断语句是否为“假”, 程序都将退出当前循环, 执行循环后的语句。

【例 4.16】break 语句的使用。

```
%break 语句的使用
for i=1:10
    if mod(i,5)==0       %如果 i 能被 5 整除, 则退出循环
        break;
    end
    n=i;
end
disp(i)
disp(n)
```

程序运行后命令窗口显示结果如下。

```
5
4
```

当 i 的值为 5 时退出 for 循环, for 循环内语句执行到 n=4。

4.6 交互控制指令

在 MATLAB 中提供了一些交互控制命令,用于让用户控制代码的执行。常用的交互控制命令如下。

1. 输入控制语句: input 命令

input 语句用于在程序运行中,用户输入变量,可以输入的变量为数值或字符串。

其调用格式如下。

- `x = input('prompt')`: 命令窗口显示提示字符串'prompt', 要求用户输入变量 x 的值。
- `x = input('prompt','s')`: 命令窗口显示提示字符串'prompt', 要求用户输入字符串型变量 x。

程序执行后,在命令窗口会给出输入变量的提示,用户根据实际需要输入变量,输入结束后按回车键结束输入。

input 语句一般用于给用户演示程序,或者给其他人使用代码,对代码输入参数不是很熟悉时使用。在一般的情况下,代码的不确定参数多数还是通过函数调用时参数的传递来实现的。

【例 4.17】input 语句的使用。

(1) 交互式的输入数值。

```
%程序用于交互式的生成不同阶数的魔术矩阵
x = input('请输入魔术矩阵的阶数'); %提示用于输入魔术矩阵的阶数
A=magic(x); %生成魔术矩阵
disp(A); %命令窗口显示魔术矩阵
```

运行代码后,命令窗口内要求输入魔术矩阵的阶数。

请输入魔术矩阵的阶数 3

```
8     1     6
3     5     7
4     9     2
```

请输入魔术矩阵的阶数 2

```
1     3
4     2
```

(2) 交互式的输入字符串。

```
%程序用于显示输入字符串的长度
x = input('请输入字符串','s'); %提示输入字符串
L=length(x); %计算字符串的长度
disp(L); %命令窗口显示字符串的长度
```

运行代码后,命令窗口内要求输入字符串。

请输入字符串 zifuchuan

```
9
```

2. 键盘输入语句: keyboard 命令

keyboard 语句主要用于程序调试或修改。当执行到此语句时,程序将停止执行,命令窗口显示提示符“K>>”,等待用户通过键盘输入操作命令。可以查看已计算的变量的情况,或者添加一些代码等。当处理完成后,用户输入 return 命令,按回车键后,程序继续运行。

keyboard 语句的调用格式为: keyboard。

【例 4.18】keyboard 语句的使用。

```
function keyboard
a=rand(1,10);
keyboard;
b=a-1;
disp(b);
end
```

程序执行后,命令窗口显示“K>>”,运行的程序停止,转入键盘控制阶段。

K>> a %输入变量 a，查看变量 a 的情况

```
a =
Columns 1 through 8
    0.8381    0.0196    0.6813    0.3795    0.8318    0.5028    0.7095    0.4289
Columns 9 through 10
    0.3046    0.1897
```

结束键盘控制，“K>>”后输入 return 命令返回程序运行。

K>> return

程序运行完毕，显示运行结果如下。

```
Columns 1 through 8
   -0.1619   -0.9804   -0.3187   -0.6205   -0.1682   -0.4972   -0.2905   -0.5711
Columns 9 through 10
   -0.6954   -0.8103
```

3. 暂停语句：pause 命令

pause 命令用于暂停运行的程序，一般用于调试过程，或者向用户显示程序的中间结果。其调用格式如下。

- pause：暂停运行中的程序，在用户按任意键后程序继续执行。
- pause(n)：暂停运行中的程序，n 秒钟之后程序继续执行。
- pause on：使程序后面的 pause 或 pause(n)指令予以执行。
- pause off：使程序后面的 pause 或 pause(n)指令不予以执行。

【例 4.19】pause 语句的使用。

(1) pause 各调用形式的使用演示。

%pause 命令的程序演示

```
x=rand;           %生成随机数 x
disp(x);          %命令窗口显示随机数 x
pause;            %程序进入暂停状态，转入命令窗口，用于让用户查看清楚程序运行的中间结果
%按任意键程序继续
clc;              %命令窗口清空
y=rand;           %生成随机数 y
disp(y);          %命令窗口显示变量 y 的值
pause(3);         %程序暂停 3 秒
pause off;        %停止程序的暂停功能
z=rand;           %生成随机数 z
pause;            %程序的暂停功能取消，不进入暂停状态
disp(z);          %显示生成的随机数 z
pause on;         %暂停功能重新开启
pause;            %进入暂停状态，按任意键继续
disp(x+y+z);      %显示 x+y+z 的值
```

运行代码，命令窗口界面上显示 x 的值如下。

```
0.6068
```

程序进入暂停状态，主界面左下角显示的程序运行状态为“Paused Press any key”，按任意键继续程序。按照程序的运行，界面清空，显示第二个随机数 y 的值。

```
0.4860
```

由于暂停功能关闭，直接显示第三个随机数 z 的值。

```
0.8913
```

执行“pause on”语句后，程序的暂停功能开启，pause 时程序进入暂停状态，按任意键继续，显示“x+y+z”的值。

```
1.9841
```

(2) pause 语句用于绘图的暂停显示。

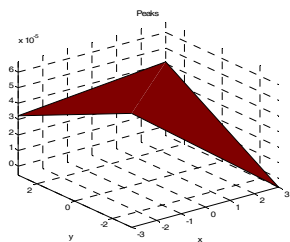
有时需要连续在同一绘图窗口显示不同状态下的图形。为方便观察，不同图形显示间可以设置暂停状态，以便于观察。

```

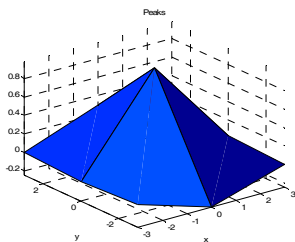
for i=2:4
    peaks(i);
    pause;
end

```

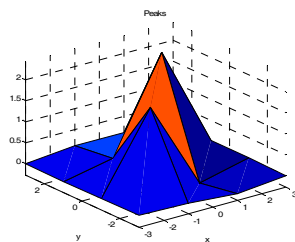
每生成一次图片，程序暂停一次，待用户观察好后，按任意键继续，如图 4.8 所示。



a (i 为 2 时的图形)



b (i 为 3 时的图形)



c (i 为 4 时的图形)

图 4.8 i 分别等于 2、3、4 时的图形

4. 指令显示控制语句：echo 命令

一般的函数文件在执行中，代码不会显示，但是如果需要查看运行了哪些代码，以便于分析时，可以使用 echo 命令。echo 语句用于控制是否显示执行的代码。如果开启显示功能，程序执行的代码将在命令窗口显示。通过这一功能可以显示查看函数文件执行的代码。

对脚本文件，其调用格式如下。

- echo on: 开启显示所有执行代码的指令。
- echo off: 关闭显示所有执行代码的指令。
- echo: 在以上两种方式之间切换，变换显示开关状态。

对函数文件，调用格式如下。

- echo function_name on: 显示函数 function_name 执行的指令。
- echo function_name off: 不显示函数 function_name 执行的指令。
- echo on all: 显示所有函数文件中执行的指令。
- echo off all: 不显示所有函数文件中执行的指令。

【例 4.20】echo 语句的使用。

(1) 脚本文件中使用。

建立如下的脚本文件。

%生成 Fibonacci 数的脚本文件

```
f(1)=1;
```

```
f(2)=1;
```

```
echo on;           %显示命令开启，此处同时也可以使用命令 echo 开启命令显示
```

```
N=10;
```

```
for n=1:N          %循环求值
```

```
    f(n+2)=f(n)+f(n+1);
```

```
end
```

```
echo off;          %关闭显示命令，此处同时也可以使用命令 echo 关闭命令显示
```

程序运行后在命令窗口将有如下显示。

```
N=10;
```

```
for n=1:N          %循环求值;
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```

        f(n+2)=f(n)+f(n+1);
        f(n+2)=f(n)+f(n+1);
        f(n+2)=f(n)+f(n+1);
        f(n+2)=f(n)+f(n+1);
        f(n+2)=f(n)+f(n+1);

```

```
end
```

```
echo off;
```

(2) 函数文件中使用。

上述脚本文件改写为函数文件 echotest2。

```
function echotest2
```

```
%生成 Fibonacci 数的函数文件
```

```
f(1)=1;
```

```
f(2)=1;
```

```
N=10;
```

```
for n=1:N      %循环求值
    f(n+2)=f(n)+f(n+1);

```

```
end
```

```
disp(f);
```

开启命令显示，执行代码。

```
>> echo echotest2 on
```

```
>> echotest2
```

```
%生成 Fibonacci 数的函数文件
```

```
f(1)=1;
```

```
f(2)=1;
```

```
N=10;
```

```
for n=1:N      %循环求值;
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
    f(n+2)=f(n)+f(n+1);
```

```
end
```

```
disp(f);
```

```
1      1      2      3      5      8      13      21      34      55      89      144
```

关闭命令显示，执行代码。

```
>> echo echotest2 off
```

```
>> echotest2
```

```
1      1      2      3      5      8      13      21      34      55      89      144
```

5. 警告语句

当程序运行出错后，程序设计中需要提供警告语句用于显示出错信息。MATLAB 提供的具有警告提示功能的语句主要有 warning 语句、error 语句、errordlg 语句，其调用格式如下。

- warning('message'): “message” 中为程序出错的提示信息，出错警告后，程序继续执行。
- error('message'): “message” 中为程序出错的提示信息，出错警告后，程序将终止运行。
- errordlg('errorstring'): “errorstring” 中为程序出错的提示信息，显示在警告对话框中，出错警告后，单击 “ok” 按钮，结束弹出式出错提示框，但代码仍会执行完毕。
- errordlg('errorstring','dlgname'): 弹出警告对话框，'dlgname' 为对话框中的标题。

【例 4.21】警告语句的使用。

(1) warning 语句。

```
function y=errortest1(x)
if any(x(:))           %判断数据是否都大于或等于 0
    warning('数据中存在小于 0 的数');
end
y=sqrt(x);
```

在命令窗口调用此函数。

```
>> x=[-2 5 6;0 9 6];           %生成矩阵 x
>> y=errortest1(x)              %调用函数 errortest1
Warning: 数据中存在小于 0 的数  %存在小于 0 的数，程序出错提示，但是不影响程序的继续运行
> In errortest1 at 3             %出错行位置信息，并提供链接到相应的出错信息行
y =
    0 + 1.4142i    2.2361            2.4495
         0         3.0000            2.4495
```

(2) error 语句。

```
function y=errortest2(x)
if any(x(:))           %判断数据是否都大于或等于 0
    error('数据中存在小于 0 的数');
end
y=sqrt(x);
```

运行代码，程序出错，停止运行。

```
>> x=[-2 5 6;0 9 6];
>> y=errortest2(x)           %运行函数
??? Error using ==> errortest2 %出错终止程序
数据中存在小于 0 的数       %出错的停止信息
```

(3) errordlg 语句。

```
function y=errortest3(x)
if any(x(:))           %判断数据是否都大于或等于 0
    errordlg('数据中存在小于 0 的数','出错信息');
end
y=sqrt(x);
```

调用函数后生成如图 4.9 所示的弹出式对话框，同时程序执行完毕。

```
>> y=errortest3(x)
y =
    Column 1
           0 +1.414213562373095e+000i
           0

    Column 2
    2.236067977499790e+000
    3.000000000000000e+000

    Column 3
    2.449489742783178e+000
    2.449489742783178e+000
```



图 4.9 “出错信息”对话框

6. 返回语句: return 命令

return 语句用于结束程序，可以强制退出某一运行的函数；而 break 只能退出循环；continue 语句只能退出当前的循环，进入下一次循环。

【例 4.22】 return 语句的使用。

```
x=[2 5 6 0 8];
for i=1:length(x)
    if x(i)==0           %当 x 中元素为 0 时退出程序
        return;
    end
    disp(x(i));
end
```

运行程序，命令窗口结果，当 x 中元素为 0 时退出程序，共显示了 x 中前 3 个非零的元素。

```
>> returntest
2
5
6
```

4.7 程序的调试

通过上述内容的学习，读者可以基本掌握程序编写的流程，可以独立写出简单的程序。相对而言，程序的写作相对容易，而写完后程序是否能保质保量地完成指定的设计任务才是关键。而对于普通用户来说，花在程序查错、调试上的时间可能远远大于程序的书写时间。因为 MATLAB 语言已为用户提供了大量的可以直接使用的函数，可以说我们在编写一段代码时，需要完全自己开发的算法部分已很少，仅是完成数据的导入、做些简单的判断，输入相应的函数完成相应的功能即可。但是，在这个过程中我们还是很容易出各种错误。出错后如何调试程序、使程序顺利完成相应的功能才是程序设计的关键，同时程序的调试需要不断地积累经验。在本节中笔者将结合多年使用 MATLAB 的心得体会，与读者探讨 MATLAB 常见的错误类型及其程序的调试方法。

4.7.1 常见错误类型

一般 MATLAB 程序中常见的错误大致可以分为 3 类。

1. 拼写错误

这类错误主要涉及函数名或者变量名拼写错误，MATLAB 运行后会提示找不到变量或函数的错误信息；MATLAB 中的一些函数需要设置固定的参数，其中参数的关键词如果拼写错误，就会提示找不到这个参数；另外一些 MATLAB 固定的关键字、流程控制语言，也会由于一时大意拼写出错，无法执行代码。但是一般 MATLAB 的关键词都会以一定颜色突出显示，当输入关键字后颜色与一般代码一样，就需要检查是否有拼写错误。同时 MATLAB 的命令是区分大小写的。一般情况下 MATLAB 自带的函数、命令都为小写状态，如果输入大写状态，则执行不了相应的任务。

这类错误是最低级的错误，也是最容易发现的，一般我们认真、仔细书写代码，可以基本避免此类错误的发生。

【例 4.23】 拼写错误的演示。

(1) 变量名的拼写错误。

```
>> error1=5;           %定义了变量 error1，并为其赋值为 5
>> y=error1+1;         %对变量 error1 计算，变量名 error1 拼写错误，错拼为“eror1”
??? Undefined function or variable 'eror1'.%程序出错，提示找不到函数或变量 eror1
```

(2) 函数名的拼写错误。

```
>> erroetest           %调用函数 erroetest，由于拼写错误，找不到函数
```

```
??? Undefined function or variable 'erroetest'.
```

找不到函数或者变量的错误，不一定是拼写错误引起的，如果函数或变量是自己生成的，保存的路径不在 MATLAB 的搜索路径下，也会出现找不到函数或变量的问题。

(3) 参数的拼写错误。

```
>> format compat                                %设置紧凑形式输出代码，但是“compact”错拼为“compat”
??? Error using ==> format                        %提示无法找到这种格式设置参数
Unknown command option.
```

(4) 大小写拼写错误。

clear 命令可用于清除 MATLAB 工作空间内的变量，但是如果在大写状态下输入，无法完成相应任务，出错提示如下。

```
>> CLEAR
??? Undefined function or variable 'CLEAR'.      %找不到函数或变量名为 CLEAR
```

2. 语法错误

这类错误主要是指一些函数的不正确使用，或者代码不符合 MATLAB 的语法规则。此类错误在代码运行后都会出现错误提示，但是有时仅根据错误提示，可能并不能解决问题，还需要整段代码的检查，因为有时前面出错的语句，可能在后面语句中才给出错误提示，但后面语句又没有出错。最典型的流程控制语句中，“for”和“end”语句需要配套使用，如果前面缺少了一个“end”，一般只会在程序的最后提示缺少“end”语句。

语法错误一般配合 MATLAB 的出错提示和函数的帮助文档，基本都能发现，采取相应措施也基本能快速解决。这类错误最容易产生，也最容易发现。

【例 4.24】常见的语法错误的演示。

(1) 函数的调用不符合格式要求。

```
>> A=magic;                                     %生成魔术阵 A，但是函数使用时没有输入参数
??? Input argument "n" is undefined.
Error in ==> magic at 13
n = floor(real(double(n(1))));
```

(2) 下标访问出错。

在通过下标访问矩阵、数组、向量中的元素时，容易出错的几个地方：超出下标的范围，下标索引从 1 开始而不是 0，且下标索引不能为负数，需要为正整数。

```
>> A=magic(3);
>> A(0)                                         %下标索引不是从 0 开始
??? Subscript indices must either be real positive integers or logicals.
>> A(1.1)                                       %下标需为正整数
??? Subscript indices must either be real positive integers or logicals.
>> A(-1)                                       %下标不能为负数
??? Subscript indices must either be real positive integers or logicals.
>> A(10)                                       %超出矩阵的下标索引上限
??? Index exceeds matrix dimensions.
```

(3) 矩阵运算维数不匹配。

在进行矩阵运算时，由于运算符（=、+、-、/、*等）两边的运算对象维数不匹配引起错误。

```
>> A=magic(3);
>> B=rand(2);
>> A*B
??? Error using ==> mtimes
Inner matrix dimensions must agree.
```

(4) 赋值时变量的大小不统一。

```
>> A=magic(3);
```



```
>> B(1)=A;           %A 为 3×3 的矩阵，无法把 A 中元素全部放到矩阵 B 的一个单元中
??? In an assignment A(I) = B, the number of elements in B and
I must be the same.
```

(5) 有时变量使用前需要预定义。

当变量需要做累加、累乘等累积运算时，变量需要预先分配一个初始值，因为最初这个变量不存在，无法进行第一次运算，会提示找不到这个变量。

```
>> for i=1:100          %1~100 的数累加求和，变量 sumi 存储累加和
    sumi=sumi+i;         %内存中不存在此变量，出错
end
??? Undefined function or variable "sumi".
```

(6) 流程控制语句格式不匹配。

3. 逻辑错误

一般存在逻辑错误的代码，可以顺利执行，也能得出结果，但是结果往往是不正确的，这类错误是隐藏最深的。程序出错与否也只能通过已知正确结果时程序运行结果的比对。

当发现代码存在逻辑错误时，需要借助于下面将要讲的调试方法来解决。

4.7.2 调试方法

程序的调试方法有很多种，总体思想是通过查看运行过程中变量值的变化来调试程序。而查看值的变化途径有以下几种。

1. 去掉变量末尾的分号

对于程序中的关键变量或者运算可能出错的变量，可以去除这些变量末尾的分号，在命令窗口将显示这些变量。但是如果这些变量数据量比较大，则通过去掉分号完全显示也不是很方便。


2. 转换函数文件为脚本文件

对于函数文件，调用完后中间变量不存储在 MATLAB 的工作空间内，这一点很不利于程序的调试，而本质上函数文件与脚本文件可以达到相同的功能，只需要把函数文件的首行调用格式暂时去除，转换为脚本文件运行。

3. 交互控制指令查看程序的运行状况

MATLAB 提供了一些交互式控制指令用于在程序运行过程中把操作权交到用户手上。其中，keyboard 语句可让程序转入键盘操作状态，此时可以查看已运行的程序生成的变量，函数文件中这些变量暂时存储在 MATLAB 的工作空间内，因而用于可以在可能出错的语句附件添加 keyboard 语句；pause 语句也可以暂停函数的运行，可以查看中间结果；echo 语句可以显示执行的代码，对于函数调用时可以准确获知函数执行了哪些代码。

4. Debugger 工具

MATLAB 提供了专门的 Debugger 工具用于程序的调试，Editor 窗口中的 Debug 菜单和标题栏中的  工具条用于程序的调试，下面详细介绍 Debugger 工具的使用。

Debug 菜单用来调试 MATLAB 的 M 文件，如图 4.10 所示，其部分菜单项功能介绍如下。

- Open M—Files when Debugging: 调试时打开 M 文件。
- Step: 逐步执行调试程序。
- Step In: 进入子函数中逐步执行调试程序。
- Step Out: 跳出子函数的调试。
- Continue: 执行调试程序，直接从一个断点处运行到下一个断点处。
- Go Until Cursor: 调试程序执行到光标所在处。

- Set/Clear Breakpoint: 设置或取消指定行的断点。
- Set/Modify Conditional Breakpoint...: 设置或修改条件断点。
- Enable/Disable Breakpoint: 显示或屏蔽指定行的断点。
- Clear Breakpoints in All Files: 清除所有文件中的断点。
- Stop if Errors/Warnings...: 在文件调试时, 出错或警告时中断程序的调试。
- Exit Debug Mode: 退出调试模式。

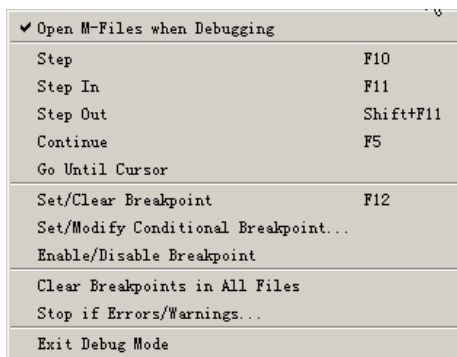



图 4.10 Editor 窗口的“Debug”菜单

【例 4.25】程序调试的演示。

(1) 光标移动到需要设置“断点”的命令行前, 设置断点的方法有快捷键“F12”、菜单 Debug → Set/Clear Breakpoint、标题栏工具 。本例中在函数文件的第 2 行和第 7 行设置了断点, 如图 4.11 所示, 断点语句前将有“小红点”标识。

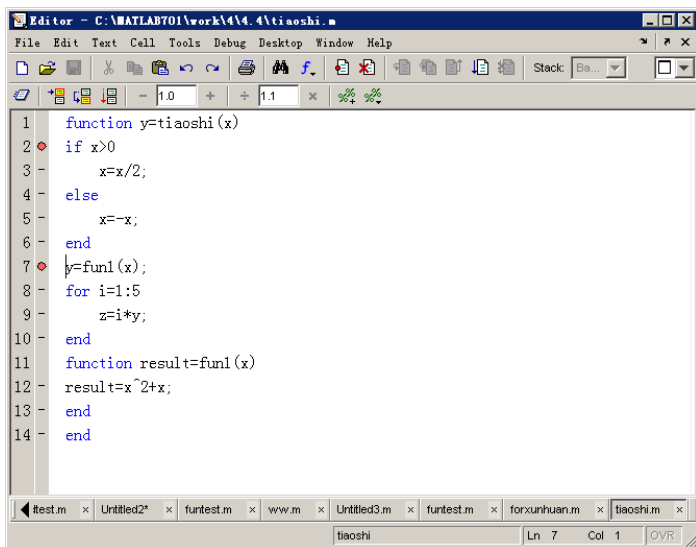


图 4.11 断点的设置

(2) 设置完断点后, 运行函数, 在断点处程序将停止执行, 进入调试模式。选择“Step”命令逐步调试执行程序。同时在调试过程中, 调试程序执行到的行前将有“绿色的箭头”标识, 如图 4.12 所示。在调试过程中可以查看函数运行的中间变量。如果认为目前所处的调试行至下一个断点处都没有问题, 可以选择“Debug”→“Continue”命令, 直接转入下一个断点的调试。默认状态下程序调试会跳过子函数, 因而如果想调试调用的子函数中的情况, 需要使用调试命令

“Step In” 和 “Step Out”。

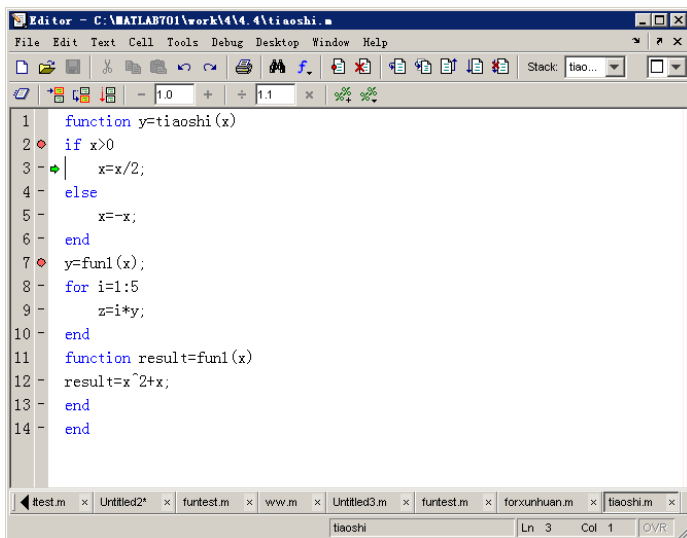


图 4.12 程序调试中

(3) 调试后发现有问题程序及时进行修改，调试完毕，退出调试模式，执行程序。

4.8 优化

在程序调试无误，可以顺利执行后，我们可能会遇到这样的问题：一方面现在的代码运行时间比较慢，是否可以进一步提高程序的效率，另一方面是否可以有办法减轻算法编程的复杂度，提高编程效率。事实上这些问题正是有一定基础的用户经常会去思考的问题，本节将介绍一些 MATLAB 代码优化的技巧，用于提高代码的效率。

4.8.1 循环的向量化

MATLAB 是基于矩阵、向量计算的，因而循环的向量化可以大大提高代码的执行效率。但是很多时候在编程时，由于受到其他语言的影响，很多用户还是会习惯性地写 for 循环，下面以一个实例比较 for 循环与向量化程序的执行效率。为测试程序执行的快慢，采用 tic 和 toc 函数计算代码运行时间。

【例 4.26】循环的向量化。

```
%使用 for 循环的代码
tic;
i=0;
for t = 0:.01:1
    i = i+1;
    y(i) = sin(t);
end
toc;

%向量化的代码
tic;
t = 0:.01:1;
y = sin(t);
toc;
```

命令窗口分别显示了循环与向量化运行的时间，程序比较简单，但是运算时间上的差异还是很大的。

```
Elapsed time is 0.015000 seconds.  
Elapsed time is 0.000000 seconds.
```

4.8.2 循环的优化

循环的向量化可以提高程序的执行效率，但是有时必须要用到循环时，也是有一些技巧可以提高循环的执行效率。

- 如果两个循环执行的次数不同，则在外循环的执行次数选择循环少的，内循环的执行次数选择循环多的，这样可以提高执行速度。
- 如果 for 循环中需要生成变量，可以预分配变量的空间，即事先确定变量的大小。

【例 4.27】 for 循环的优化。

```
%% for 循环的优化  
tic;  
clear  
for i=1:300  
    for j=1:100  
        a(i,j)=rand;  
    end  
end  
toc;  
%% 外循环设置循环次数多的循环  
tic;  
clear  
for j=1:100  
    for i=1:300  
        a(i,j)=rand;  
    end  
end  
toc;  
  
%% 循环前变量预定义  
tic;  
clear  
a=zeros(300,100);  
for j=1:100  
    for i=1:300  
        a(i,j)=rand;  
    end  
end  
toc;
```

程序执行后三段 for 循环代码的运行时间如下。

```
Elapsed time is 8.674000 seconds.  
Elapsed time is 0.063000 seconds.  
Elapsed time is 0.016000 seconds.
```

其中以变量预定义，外层循环次数少的代码运行效率最高。

4.8.3 M 文件分析

Editor 窗口中的 Tools 菜单用于 M 文件的分析，如图 4.13 所示，其中，

- Check Code with M-Lint：用于检查 M 文件代码逐行检查，给出代码的错误信息。
- Show Dependency Report：列出当前 M 文件调用的子程序。
- Open Profiler：生成程序执行效率的分析文件。

Check Code with M-Lint
Show Dependency Report
Open Profiler

图 4.13 Editor 窗口中的“Tools”菜单

【例 4.28】M 文件分析。

以下面一段脚本文件为例，演示 M 文件分析各功能的使用。

```
A=rand(3);
x=mean(A)
y=mean(A,2)
z1=mean(mean(A))
z2=mean(A(:))
```

选择“Tools”→“Check Code with M-Lint”命令，对当前的 M 文件进行分析，打开如图 4.14 所示的窗口。给出 M 文件的代码检查信息，本例中给出的信息为代码后未加分号。

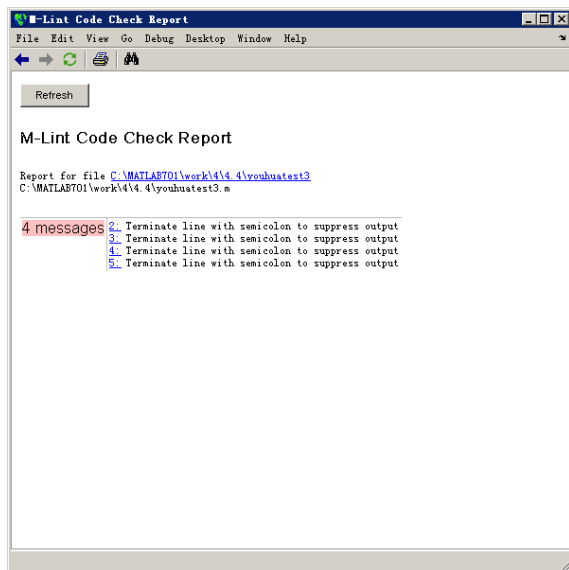


图 4.14 “M-Lint Code Check Report”窗口

选择“Tools”→“Show Dependency Report”命令，对当前的 M 文件进行分析，打开如图 4.15 所示的窗口，显示当前 M 文件所调用的函数。

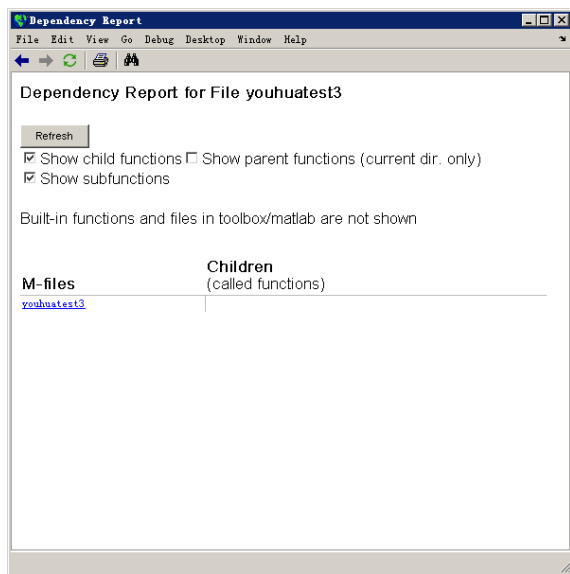


图 4.15 “Dependency Report” 窗口

选择 “Tools” → “Open Profiler” 命令，将打开如图 4.16 所示的代码效率分析窗口，单击窗口中的 “Start Profiling” 按钮，将对当前 M 文件运行花费的时间分析。根据分析，可以重点在花费时间最多的语句代码块上尝试提高速率。

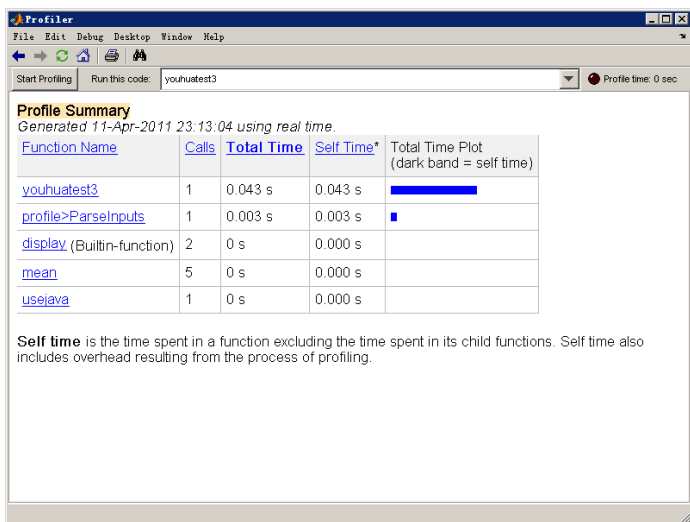


图 4.16 代码效率分析窗口

4.8.4 提高编程效率的小技巧

(1) 线性索引坐标的使用。一般在矩阵操作中，我们使用的比较多的是行列坐标，但是 MATLAB 中本身元素是按列存储的，线性坐标的使用可以很方便地提取元素中的值。

【例 4.29】 线性索引坐标的使用。

```
>> A=rand(3)           %生成随机矩阵 A
A =
    0.9141    0.9823    0.0152
    0.0431    0.4087    0.4722
    0.2960    0.1899    0.8476
```

```

>> mean(A)                %计算矩阵 A 的行均值
ans =
    0.4177    0.5269    0.4450
>> mean(A,2)              %计算矩阵 A 的列均值
ans =
    0.6372
    0.3080
    0.4445
>> mean(A(:))             %通过线性索引坐标可以直接提取矩阵 A 的所有元素，计算其均值
ans =
    0.4632
>> [maxA,c]=max(A(:))     %求取矩阵 A 的最大元素，返回最大值和线性坐标
maxA =
    0.9823
c =
     4
>> [i,j]=ind2sub(c,size(A)) %线性坐标转换为行列坐标
i =
     3
j =
     1

```

(2) “:” 和 end 的使用。“:” 在矩阵中可以代替一行、一列，或者全部元素，而 end 可以提取最后的元素、指定行、指定列到最后的元素。“:” 和 end 语句的使用让编写程序时不用关心矩阵具体的大小。

【例 4.30】“:” 和 end 的使用。

```

A=magic(3);                %生成奇异矩阵 A
A(:,1)=A(:,1)+1;          %提取第 1 列的元素并对其进行加 1 操作
A(2,:)=2*A(2,:);          %提取第 2 行的元素并对其进行乘 2 操作
A(:)=A(:)+3;              %提取矩阵 A 中的所有元素
A(2,2:end)=A(2,2:end)+1;  %提取矩阵 A 中第 2 行中第 2 列至结尾的数据进行加 1 操作

```

(3) 调试程序时，经常希望暂时屏蔽部分程序段，可以将这部分标识为注释。具体的操作：选定待屏蔽的内容，选择“Comment”命令。恢复为程序只要选定注释代码并选择“Uncomment”命令即可。

(4) 程序进入死循环，想终止程序，但是不想结束 MATLAB，可以通过“Ctrl+Break”组合键强制终止程序。

(5) 按“Ctrl+I”组合键可以让代码按照编程习惯自动地对齐。

4.9 本章小结

本章主要介绍了 MATLAB 程序设计的基本知识。首先，简单介绍了 MATLAB 程序设计概述类的常识；然后，介绍了程序设计的重要组成元素、脚本文件和函数文件、常量和变量；接下来介绍了程序结构及流程控制的基本知识，这部分内容与一般的高级语言类似，读者在学习的过程中可以触类旁通；紧接下来介绍了交互控制指令，使读者可以更好地控制代码的执行；最后，介绍了程序调试与优化的相关内容，将有助于读者解决实际编写代码中遇到的各种问题。

通过本章的学习，读者应该能掌握 MATLAB 程序设计基本的语法规则，应该尝试编写一些简单的程序文件，并在程序出错后能尝试调试修改程序，同时进一步完善程序，使其效率更高。



第5章

图形处理

MATLAB 提供了强大的数值计算处理功能，而数值结果如果能以图形化的方式显示，将有利于在实际问题中直观地理解计算的结果。本章主要介绍 MATLAB 图形处理的相关知识，涉及基本的绘图处理，二维、三维图形的绘制，以及图形编辑处理技巧。MATLAB 的图形处理功能强大，且操作简单，可以在图像中直接通过鼠标和键盘操作完成，同时也可以通过设计程序完成相应的功能。

通过本章节的学习将使读者掌握 MATLAB 数据图形化的方法，绘制各类型的图形，并根据实际需要编辑调整绘制的图形，力求做出美观、准确的图形。

5.1 基本的绘图处理

本节将介绍 MATLAB 基本的绘图处理。其中 5.1.1~5.1.5 小节主要从函数命令的角度向用户介绍绘图处理，包括绘图的基本函数、坐标轴、标注的设置等，而第 5.1.6 小节将在 MATLAB 的图形窗口演示如何利用界面中编辑选项，完成图形的编辑。

5.1.1 常用函数

MATLAB 绘图的常用函数主要有以下几种。

1. plot(): 二维图形绘制

- plot(y): 对于只含一个输入参数的 plot 函数，如果输入参数 y 为向量，则以该参数为纵坐标，横坐标从 1 开始至与向量的长度相等；如果输入参数 y 是矩阵时，则按列绘制每列元素的曲线，每条曲线的纵坐标为该列上的元素值，横坐标从 1 开始，与元素的行坐标对应，曲线条数等于输入参数矩阵的列数，多条曲线默认状态下通过颜色区别。
- plot(x,y): 对于含有两个输入参数的 plot 函数，如果 x 是向量， y 也为向量，则向量 x 、 y 的长度必须相同；如果 x 为向量时， y 为矩阵，则矩阵 y 必须有一维大小与向量长度相等，将以 x 为横坐标，绘制出多条不同颜色的曲线，曲线条数等于矩阵 y 的另一维的大小；如果 x 、 y 是同维矩阵，则分别以矩阵 x 、 y 对应列元素为横、纵坐标，绘制曲线，曲线条数等于矩阵的列数，不同的曲线默认状态下会以不同的颜色区别。
- plot(x1,y1,x2,y2,...): 对于含有多个输入参数的 plot 函数， $x1$ 和 $y1$ 、 $x2$ 和 $y2$ 分别配对，即以 $x1$ 为横坐标数据时， $y1$ 为相应的纵坐标，以 $x2$ 为横坐标数据时， $y2$ 为相应的纵坐标，以此类推，要求配对的向量长度相等，但是组间向量可以不相等，最终可以在同一图形窗口内绘制出多条曲线。
- plot(x1,y1,LineSpec): 用于对图形的线型、数据点的样式、颜色进行控制，LineSpec 为控制线型、点型、颜色的字符串。MATLAB 7.0 中线型、点型、颜色的控制符分别如表 5.1、

表 5.2 和表 5.3 所示。3 个控制符连为字符串对图形样式控制，线型、点型、颜色的控制符的位置对结果没有影响，可以缺省任何一个或多个参数。例如，“r-.”表示红色点画线，“y--p”表示黄色虚线并用五角星标记数据点。如果使用 plot 函数的数据参数为矩阵数据绘制多条曲线时，设置了图形样式，各曲线的样式将统一，一般不建议这样操作。

- plot(x1,y1,'PropertyName',PropertyValue)：对绘制的图形属性进一步设置。其中 PropertyName 为曲线的属性名称，PropertyValue 为属性的值，属性和属性值需要成对出现，且不同属性之间没有前后顺序关系。如表 5.1~表 5.4 所示列出了常用的属性及其说明。

表 5.1 plot 函数线型控制符

线条样式	控制符
实线	-
点线	:
虚线	--
点画线	-.

表 5.2 plot 函数点型控制符

数据点样式	控制符
点号	.
十字符	+
*号	*
叉号	×
空心圆	o
正方形	s
五角星	p
菱形	d
六角星	h
上三角	^
下三角	∨
左三角	<
右三角	>

表 5.3 plot 函数颜色控制符

颜色属性	控制符
红色	r
粉红	m
绿色	g
青色	c
蓝色	b
白色	w
黄色	y
黑色	k

表 5.4 plot 函数常用属性

属性名	描述
LineWidth	设置线的宽度
MarkerSize	设置标记点的大小
MarkerEdgeColor	设置标记点的边缘颜色
MarkerFaceColor	设置标记点的填充颜色

【例 5.1】plot()函数的使用。

```
y1=randperm(10);           %生成10个随机数的向量
plot(y1);                   %绘制向量的二维图形
```

运行后显示的图形如图 5.1 所示，横坐标为 1~10，纵坐标为向量的值。

```
y2=rand(3,4);           %生成 3×4 的矩阵
plot(y2);                %绘制矩阵的二维图形
```

运行后显示的图形如图 5.12 所示，矩阵 y_2 每列元素生成一条曲线，共 4 条曲线。

```
x1=0:pi/100:2*pi;       %x 为向量
y1=0.2*cos(4*pi*x1);     %y 为与 x 长度相同的向量
plot(x1,y1)              %绘制输入参数均为向量的二维图形
plot(x1',y1)              %plot 绘图对向量为列向量或者行向量没有严格要求
plot(x1,y1')              %向量转置后绘制的图形不变
```

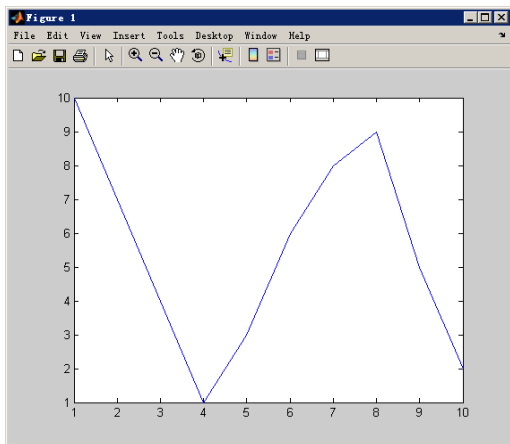


图 5.1 plot(y)函数绘图 (y 为向量)

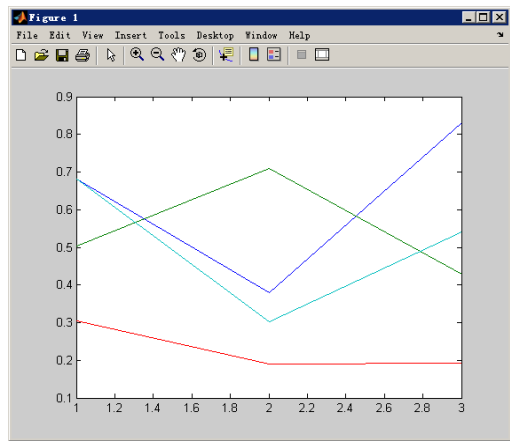


图 5.2 plot(y)函数绘图 (y 为矩阵)

运行后 3 个 plot()函数均显示如图 5.3 所示的图形。使用 plot()函数绘图时，如果对曲线的线条没有设置，默认状态下为折线图 (图 5.1、图 5.2)，但是如果数据间隔足够小，折线图看上去接近曲线图 (如图 5.3 所示)。

```
x1=0:pi/100:2*pi;       %x 为向量
for i=1:4
    y1(i,:)=i*cos(pi*x1); %y 为一维与 x 长度相同的矩阵
end
plot(x1,y1)              %绘制二维图形，曲线数等于矩阵 y 另一维的大小
```

运行上述程序，生成如图 5.4 所示的图形。

```
x=[1 2 3;4 5 6;7 8 9];
y=rand(3);
plot(x,y);
```

运行上述程序，生成如图 5.5 所示的图形。

```
x1=0:pi/100:2*pi;
x2=0:pi/100:pi;
plot(x1,sin(x1),x2,cos(x2));
```

运行上述程序，显示如图 5.6 所示的图形，生成了两条曲线。

```
x=0:pi/10:2*pi;
plot(x,sin(x),'ro-',x,cos(x),'--p') %绘图设置图形不同的线型、点型、颜色
```

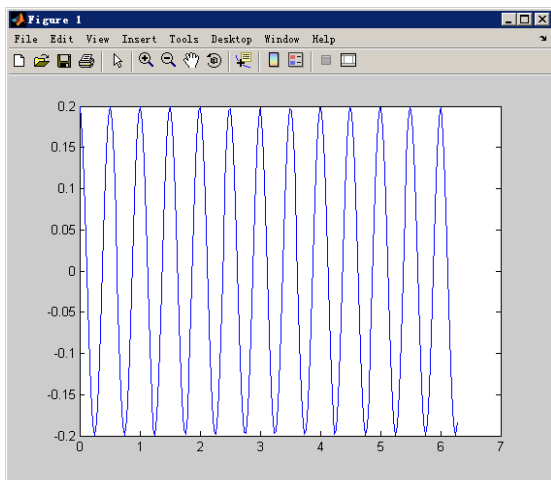


图 5.3 plot(x,y)函数绘图 (x,y 为向量)

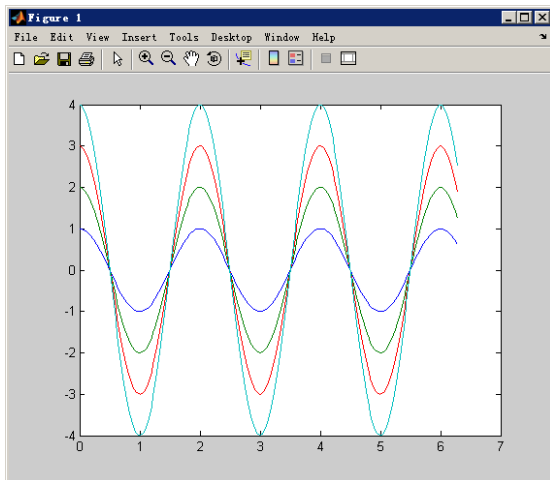


图 5.4 plot(x,y)函数绘图 (x 为向量,y 为矩阵)

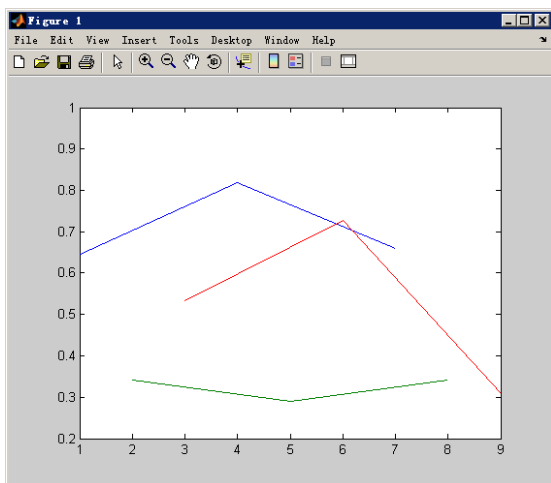


图 5.5 plot(x,y)函数绘图 (x、y 为相同大小的矩阵)

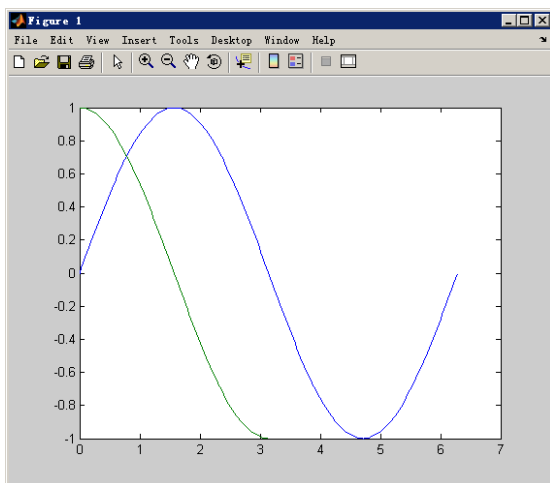


图 5.6 plot(x1,y1,x2,y2)函数绘图

运行上述程序，显示如图 5.7 所示的图形，生成了不同线型、点型、颜色的曲线。

```
x=0:pi/20:pi;
y=sin(4*x);
plot(x,y,'ro--','LineWidth',3,... %设置线的宽度为 3
'MarkerEdgeColor','k',... %设置标记点边缘颜色为黑色
'MarkerFaceColor','y',... %设置标记点填充颜色为黄色
'MarkerSize',10) %设置标记点的尺寸为 10
```

运行上述程序，显示如图 5.8 所示的图形，曲线设置了线宽、标记点的尺寸、颜色。

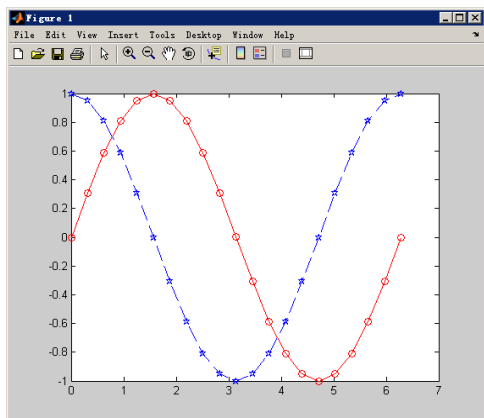


图 5.7 plot()函数绘图设置图形的线型、点型、颜色

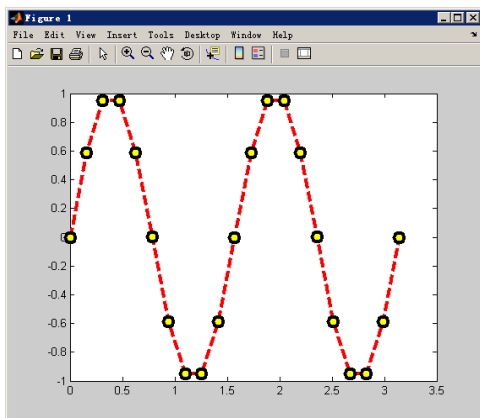


图 5.8 plot()函数属性设置

2. plotyy(): 双 y 轴图形绘制

在实际中, 如果两组数据的数据范围相差较大, 而又希望放在同一图形中比较分析, 可以绘制双 y 轴图形。其调用格式如下。

- `plotyy(x1,y1,x2,y2)`: 其中 $x1$ 、 $y1$ 和 $x2$ 、 $y2$ 分别为一组曲线。绘制的图形横坐标标度相同, 纵坐标左右各有一个标度, 左纵坐标对应 $x1$ 、 $y1$ 数据对, 右纵坐标对应 $x2$ 、 $y2$ 数据对。
- `plotyy(x1,y1,x2,y2,'function')`: 以 “function” 函数形式绘制双 y 轴形式, function 可为任意可接收 `function(x,y)` 形式绘图的函数, 可以是 MATLAB 系统自带的 `plot`、`semilogx`、`semilogy`、`loglog`、`stem` 函数或者用户自定义的函数。
- `plotyy(x1,y1,x2,y2,'function1','function2')`: 曲线 $x1$ 、 $y1$ 以 “function1” 形式绘制, 曲线 $x2$ 、 $y2$ 以 “function2” 形式绘制。

【例 5.2】`lotyy()` 函数的使用。

```
%% plotyy() 函数调用格式 1 的使用
x1=0:pi/100:2*pi;
plotyy(x1,sin(x1),x1,100*cos(x1));           %双 y 轴图形绘制
%% plotyy() 函数调用格式 2 的使用
plotyy(x1,sin(x1),x1,100*sin(x1).*x1,'loglog'); %以 loglog() 函数形式绘制双 y 轴图形
%% plotyy() 函数调用格式 3 的使用
plotyy(x1,sin(x1),x1,100*cos(x1),'loglog','plot'); %分别以 loglog() 和 plot() 函数形式绘制双 y 轴图形
```

执行上述程序, 使用函数 `plotyy()` 的 3 种调用格式分别生成的图形如图 5.9、图 5.10 和图 5.11 所示。

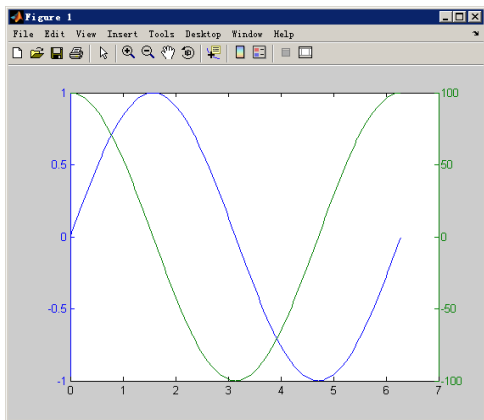


图 5.9 plotyy()函数绘制的图形 1

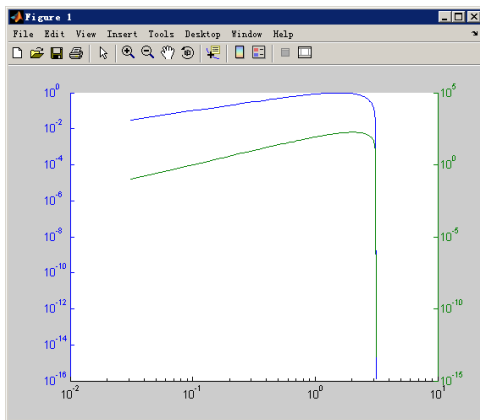


图 5.10 plotyy()函数绘制的图形 2

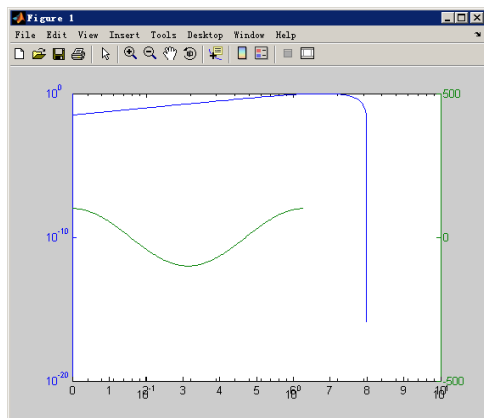


图 5.11 plotyy()函数绘制的图形 3

3. loglog(): 对数坐标图形的绘图

函数 loglog() 用于 x、y 轴均为对数的坐标系绘图。函数调用格式如下：

- loglog(y)
- loglog(x1,y1)
- loglog(x,y,LineStyle)
- loglog(x,y,'PropertyName',PropertyValue)

各调用格式的用法与 plot() 函数类似，在此不再展开叙述，读者可以参考 plot() 函数的使用。

【例 5.3】 loglog() 函数的使用。

```
x=1:10:100;
loglog(x,exp(2*x));
```

执行上述程序，生成如图 5.12 所示的图形。

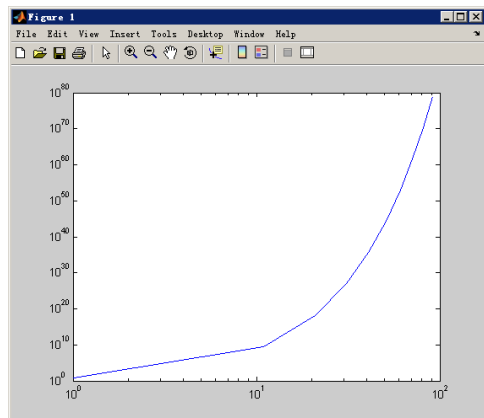


图 5.12 loglog()函数的使用

4. semilogx() / semilogy(): 半对数坐标图形的绘制

semilogx(): 绘制 x 轴为对数坐标，y 轴为线性坐标的二维图形。

semilogy(): 绘制 y 轴为对数坐标，x 轴为线性坐标的二维图形。

函数的具体调用格式也同 plot 函数，在此不再详细展开叙述。

【例 5.4】 semilogx() / semilogy() 函数的使用。

```
x=1:10:100;
semilogx(x,exp(2*x));
```

%绘制 x 轴为对数坐标，y 轴为线性坐标的二维图形

`semilogy(x,exp(2*x));` %绘制 y 轴为对数坐标, x 轴为线性坐标的二维图形
 执行上述程序,生成如图 5.13 和 5.14 所示的图形。

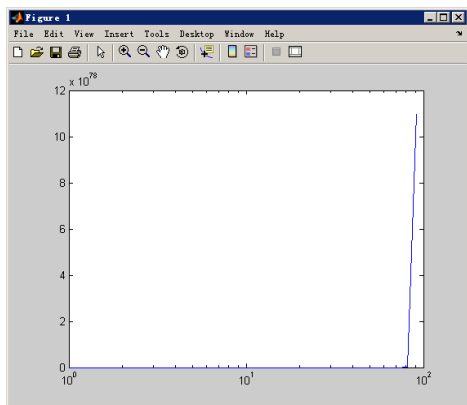


图 5.13 semilogx()函数的使用

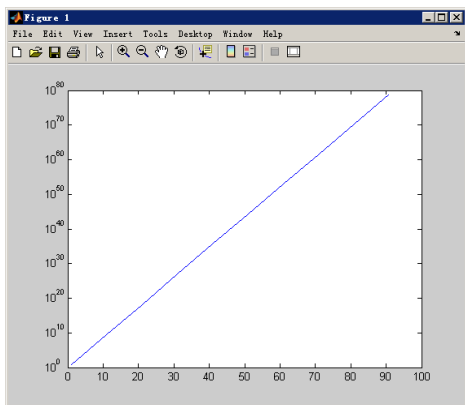


图 5.14 semilogy()函数的使用

5. fplot(): 符号函数的绘制

上面介绍的函数主要用于离散数据的二维图形绘制,而函数 `fplot()` 可以根据函数的表达式自动调整自变量的范围,无须给函数赋值,直接生成能反映函数变化规律的图形,在函数变化快的区域,采用小的间隔,否则采用大的坐标间,使绘制的图形计算量与时间最小,而又能尽可能精确反映图形的变化。`fplot()` 函数一般在对横坐标取值间隔没有明确要求,仅查看函数的大致变化规律的情况下使用。函数的调用格式如下。

- `fplot('function',limits)`: 在指定的坐标值范围 `limits` 内绘制函数 `function` 的图形。其中 `limits` 是指定 `x` 轴范围的向量 `[xmin xmax]` 或同时指定 `x` 轴和 `y` 轴范围的向量 `[xmin xmax ymin ymax]`; 函数 `function` 必须是一个包含 `y=f(x)` 的 `M` 文件或包含变量 `x`,且能用函数 `eval` 计算的字符串。
- `fplot('function',limits,LineSpec)`: `LineSpec` 参数设置图形的线型、数据点的样式、颜色。
- `fplot('function',limits,err)`: 绘制函数 `function` 时允许的相对误差值为 `err`,默认状态下相对误差的值为 $2e-3$ 。

【例 5.5】fplot()函数的使用。

`fplot('sin(x)*cos(x)^3',[0,2*pi]);` %指定 x 轴范围
`fplot('sin(x)*cos(x)^3',[0,2*pi,0,0.1]);` %同时指定 x 轴和 y 轴的范围
 执行上述程序,显示如图 5.15 所示的图形。

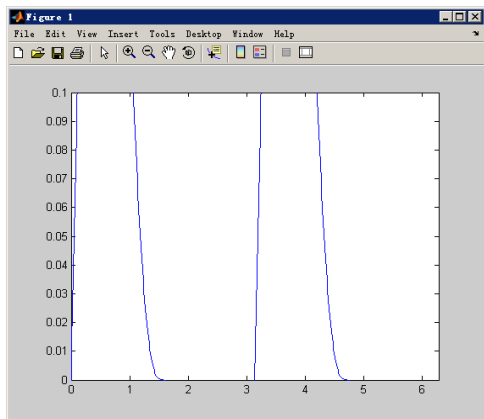
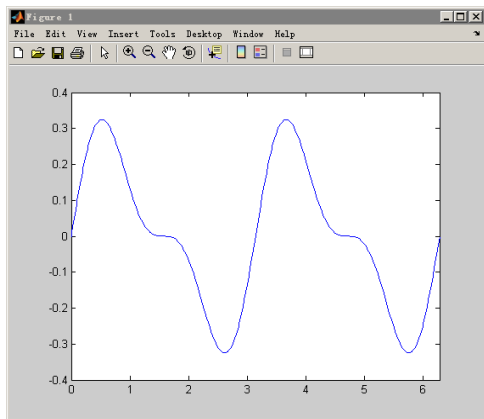


图 5.15 fplot()函数的使用

6. ezplot(): 符号函数的绘制

ezplot()函数与 fplot 函数的功能基本类似,可以方便地绘制表达式或函数的图形。与 fplot 函数不同的是函数的表达式显示在图形的上方,同时对坐标轴可不加任何限制做图。其调用格式如下。

- ezplot(f): 绘制表达式或函数的图形,默认 x 轴的范围是 $[-2\pi, 2\pi]$ 。
- ezplot(f,[min,max]): 设置绘图时 x 轴的范围。
- ezplot(f,[xmin,xmax,ymin,ymax]): 同时设置绘图时 x 轴和 y 轴的 range。

【例 5.6】 ezplot()函数的使用。

```
ezplot('x^2+sin(x)');
```

执行上述程序,生成如图 5.16 所示的图形。

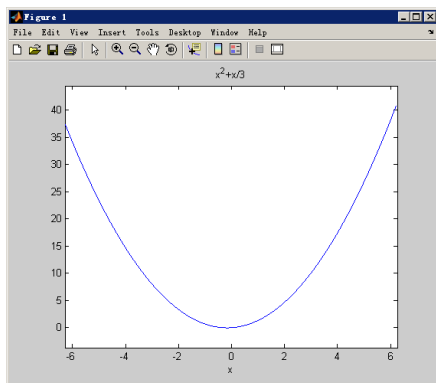


图 5.16 ezplot()函数的使用

7. Grid(): 图形栅格的控制

函数 grid 用于在二维或三维图形上控制坐标轴的栅格显示。其具体的调用格式如下。

- grid on: 给当前的坐标轴添加栅格。
- grid off: 取消当前的坐标轴中的栅格。
- grid: 图形栅格显示状态的切换。

【例 5.7】 grid()函数的使用。

```
x=[0:2*pi];
plot(x,sin(x));
grid on;           %给当前坐标轴的图形添加栅格
grid off;          %去除当前坐标轴的图形栅格显示
grid;              %当前坐标轴不显示栅格, grid 命令切换至显示状态
```

执行上述程序,显示如图 5.17 所示的图形。

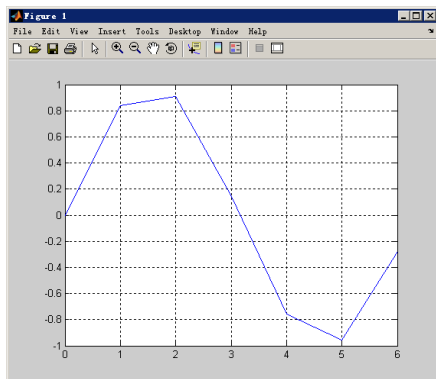


图 5.17 grid()函数的使用, 显示栅格状态

8. Hold(): 图形保持的控制

在图形窗口生成图形后, 再生成下一幅图形的时候, 会覆盖原图形, 如果想保留原图形可以使用 hold 函数用于图形保持控制。函数 hold 的调用格式及用法如下。

- hold on: 当前图形窗口中的图形处于保持状态, 后续图形叠加在原有图形上。
- hold off: 关闭图形窗口中的图形保持状态, 后续图形覆盖原有图形。
- hold: 在 hold on 与 hold off 之间转换。即在叠加与覆盖图形之间切换。

同时, 函数 ishold 用于测试图形的保持状态, 返回值“1”表示图形处于叠加状态, “0”表示图形处于覆盖状态。

【例 5.8】hold()函数的使用。

```
x=[0:2*pi];
plot(x,sin(x));
hold on           %开启图形的保持功能
plot(x,cos(x),'r-'); %新绘制的图形叠加在原有图形上
hold off         %关闭图形的保持功能
plot(x,tan(x),'b'); %新绘制的图形叠加在原有图形上
```

执行上述程序, 图形保持状态下的图形如图 5.18 所示。

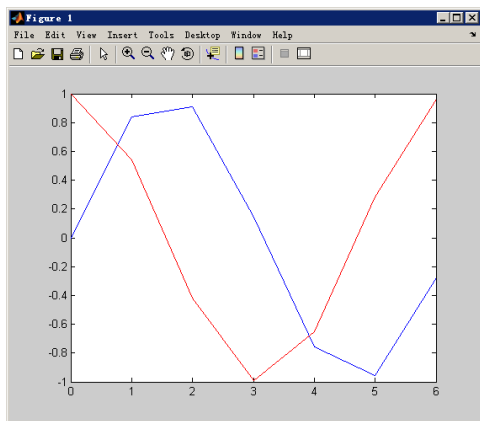


图 5.18 hold()函数的使用

9. ginput(): 读点函数

函数 ginput()用于交互式从 MATLAB 绘制的图形中读取点的坐标, 其调用格式如下。

- [x,y]=ginput(n): 用于交互式的通过鼠标读取图形中的点, 返回点的横纵坐标值, 其中 x 为点的横坐标值, y 为点的纵坐标值, 输入参数 n 为选择的点的个数, 可以按“Enter”键提前结束读点操作。
- [x,y]=ginput: 可以无限地读取图形中点的坐标直到按下“Enter”键。
- [x,y,button]=ginput: button 值返回读点时的鼠标操作, 其中“1”代表按下鼠标左键读点, “2”代表按下鼠标中键读点, “3”代表按下鼠标右键读点, 通过不同鼠标按键的区别, 可以对读取点进行分类。

【例 5.9】ginput()函数的使用。

```
x1=0:pi/100:2*pi;
plot(x1,sin(x1));
n=10;
[x,y]=ginput(n); %读取图形中的 10 个坐标点
[x,y]=ginput;    %读取图形中的任意多个点
```


`[x,y,button]=ginput;` %返回读取点的鼠标操作
执行上述代码, 读点操作如图 5.19 所示交互式地进行。

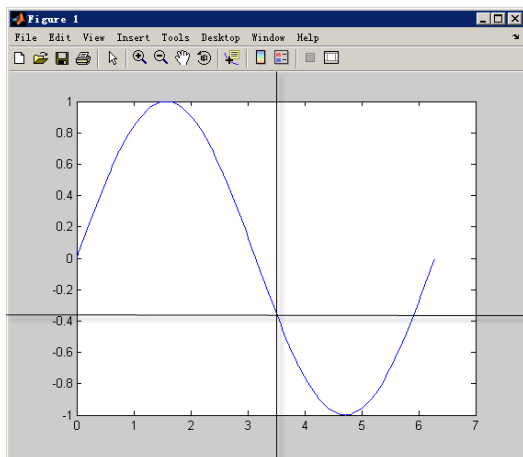


图 5.19 ginput()函数的使用

10. zoom(): 图形缩放

zoom()函数用于对二维图形的缩放控制, 放大或缩小会相应地改变坐标轴范围。其调用格式如下。

- zoom on: 打开图形的缩放功能, 通过单击鼠标, 可以放大图形, 放大图形后要想缩小图形, 需要按住“Shift”键, 再单击鼠标键。
- zoom off: 关闭交互式图形缩放功能。
- zoom out: 将缩放后的图形恢复到原始状态。
- zoom reset: 将当前图形的状态作为“原始态”, 以后使用 zoom out 图形恢复到此状态。
- zoom: 用于切换缩放的状态。
- zoom xon: 仅对 x 轴进行图形缩放。
- zoom yon: 仅对 y 轴进行图形缩放。

11. saveas(): 图形保存

saveas()函数可以按照指定的格式保存图形, 其使用的格式如下。

- saveas(gca,'filename','fileformat'): 其中 gca 指明保存当前的图形对象, filename 为图形的文件名, 可以为绝对路径下的文件名, 或者直接为文件名, 保存至当前路径下, fileformat 保存的图形的文件类型, 包括常见的多种图形格式: fig、eps、emf、png、jpg、tif 等常用的图片格式类型, 其中 fig 为 Matlab 默认图形文件类型, 保存为该格式后, 下次打开后可以直接编辑修改。

12. print(): 图形打印

print()函数可用于打印图形窗口的图形, 必须紧跟在函数 plot()后使用, 其调用格式如下。

- print filename: 命令中“filename”为打印的图形文件文件名。
- print -dformat: 命令中“dformat”为打印的图形文件的存储格式。
- print ...-option: 命令中“option”设置打印的图形文件的参数。

5.1.2 MATLAB 图形窗口

通过 5.1.1 小节的学习, 可以发现通过 MATLAB 绘制的图形都显示在图形窗口。图形窗口主要用于图形的显示和编辑。本小节中将介绍图像窗口创建、图形句柄及图形窗口常用的操作命令。

1. 图形窗口的创建

图形窗口的创建可以通过选择“File”→“New”→“Figure”命令来创建，同时 MATLAB 中也提供了函数 `figure()` 用于创建图形窗口。`figure()` 函数的调用格式如下。

- `figure`: 创建一个新的图形窗口，窗口各对象的属性值取默认状态。
- `figure(n)`: 创建图形窗口，其中 `n` 表示第 `n` 个窗口，即 `Figure n`。
- `figure(h)`: 其中 `h` 为窗口的句柄，则句柄 `h` 的窗口将作为当前窗口，绘制的图形将显示在当前窗口中。
- `figure('PropertyName',PropertyValue,...)` 设置创建窗口的指定属性 `PropertyName` 的属性值 `PropertyValue`。

【例 5.10】图形窗口的创建。

```
figure;
```

在命令窗口运行上述命令，生成如图 5.20 所示的空图形窗口，图形窗口的属性都为默认状态下的属性。此时新创建的图形窗口为当前的图形窗口，新绘制的图形将在此图形窗口显示。默认状态下新创建的图形窗口按照数字“1, 2, 3...”的顺序依次命名，即第一个窗口为 `Figure 1`，第二个窗口为 `Figure 2`，以此类推，但 `figure(n)` 可以生成指定数字的图形窗口 `Figure n`。

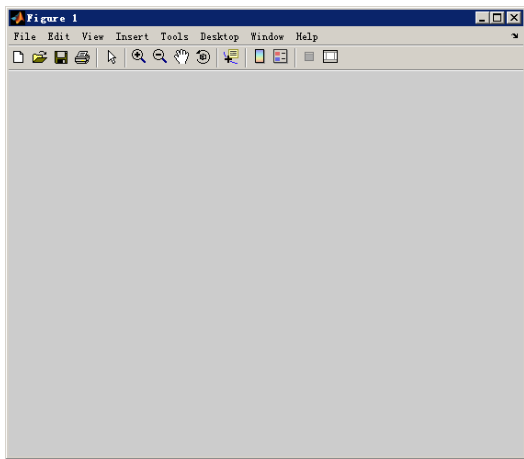


图 5.20 图形窗口的创建

2. 图形句柄

不同的图形可以显示在不同的图形窗口，在对不同图形窗口的属性设置时，首先需要让程序明确对哪个窗口的哪个对象进行操作，图形句柄即用于不同图形窗口对象的标识。通过图形句柄，可以对该句柄下标识的图形窗口对象的属性进行设置。图形句柄由 MATLAB 自动生成，在每新建一个图形窗口时都有相应的句柄与其对应，用户不能更改，但可以获取不同窗口的句柄值。

(1) 获取图形对象的句柄值。

在创建图形窗口时即可通过函数 `h=figure`，返回新创建的图形窗口的句柄到变量 `h` 中。

函数 `gcf` 可获取当前图形窗口的句柄，当前图形窗口即将显示新绘制图形的窗口；函数 `gca` 可获得当前坐标轴对象的句柄；函数 `gco` 可以获得当前对象的句柄，当前对象即用户鼠标最后单击的对象。上述 3 个函数的调用格式如下。

- `h = gcf`: 其中变量 `h` 返回当前图形窗口的句柄。
- `h = gca`: 其中变量 `h` 返回当前坐标轴对象的句柄。
- `h = gco`: 其中变量 `h` 返回当前对象的句柄。

函数 `findobj()` 可通过图形对象的属性值查找相应图形对象的句柄。其调用格式如下。

- `h=findobj`: 返回根对象与其所有子对象的句柄值。

- `h=findobj('PropertyName',PropertyValue)`: 查找属性 `PropertyName` 的属性值为 `PropertyValue` 的图形对象的句柄值, 返回到变量 `h` 中。

仅获得函数句柄并没有很大的意义, 一般在获得句柄后可以获取、设置句柄标识的图形对象的属性。

(2) 使用图形句柄进行属性获取和设置。

函数 `get()` 用于获取指定句柄下的图形对象的属性。其常用的调用格式如下。

- `get(h)`: 返回由句柄 `h` 标识的图形对象的所有属性及其属性值。
- `get(h,'PropertyName')`: 返回由句柄 `h` 标识的图形对象的属性 `PropertyName` 的当前属性值。
- `a=get(h)`: 返回由句柄 `h` 所标识的图形对象的属性结构体数组, 其中该结构体的各域名为对象的属性名, 域的值相应属性的当前属性值。

函数 `set` 用于设置指定句柄下的图形对象的属性。其常用的调用格式如下。

- `a=set(h)`: 返回句柄 `h` 所标识的图形对象的所有属性名及其属性值, 其中 `a` 为结构体数组, 该结构体的各域名为对象的属性名, 域的值相应属性的当前属性值, 属性值中用 `{}` 括起来的表示默认值。
- `set(h,'PropertyName',PropertyValue)`: 设置由句柄 `h` 标识的图形对象的属性 `PropertyName` 的当前属性值为 `PropertyValue`。

(3) 图形窗口的属性。

窗口位置和大小属性设置。

- `Position` 属性: 通过向量 `[left,bottom,width,height]` 定义图形窗口的位置与大小, 其中 `lef`、`bottom` 为窗口左下角的坐标, `width` 为图形窗口的宽, `height` 为图形窗口的高。
- `Units` 属性: `Position` 属性中参数的单位, 可以设置的单位有 `inches` (英寸)、`centimeters` (厘米)、`normalized` (标准化单位)、`points` (点)、`pixels` (像素) 和 `characters` (字符), 默认状态下为 `pixels`。

窗口显示属性的设置。

- `Color` 属性: 设置窗口的背景颜色, 通过相应颜色的参数设置。
- `Menubar` 属性: 设置图形窗口菜单栏与工具栏的显示与否, 属性值 `none` 代表不显示、属性值 `figure` 代表显示, 默认状态下图形窗口显示菜单栏与标题栏。
- `NumberTitle` 属性: 设置标题栏是否显示图形窗口的编号 (Figure Number), 默认状态下属性值为 `"on"`, 表示显示状态; 属性值设置为 `"off"` 即表示不显示。
- `Name` 属性: 图形窗口名称的显示, 可以为任意字符串, 默认状态下为空, 设置完成后将在标题栏 Figure Number 后显示 `Name` 字符串。
- `Resize` 属性: 设置图形窗口的大小是否可以通过鼠标改变, 默认状态下属性值为 `"on"`, 可以通过鼠标改变窗口大小; 当属性值设置为 `"off"` 时表示窗口固定, 不可以通过鼠标改变窗口大小。
- `Visible` 属性: 确定图形窗口是否可见, 默认状态下可见, 属性值为 `"on"`, 属性值设置为 `"off"` 图形窗口不可见。

图形窗口的其他一些属性, 不是很常用, 如有需要读者可以自行尝试改变属性值, 体会属性的设置方法。

【例 5.11】 图形句柄的使用。

```
h=figure;
```

%新建图形窗口, 并返回图形窗口的句柄到变量 `h` 中

```

get(gcf,'position')           %获取图形窗口的位罦信息
get(h,'units')               %获取图形窗口的位罦信息
set(gcf,'position',[320 260 300 300]) %设置图形窗口的位罦信息

```

3. 图形窗口的常用操作命令

(1) clf / cla: 清除命令。

- clf: 清除当前图形窗口的图形。
- cla: 清除当前图形窗口坐标轴内的图形, 保留坐标轴。

(2) delete(): 删除命令。

- delete()函数用于删除文件或图形对象, 将永久性地删除文件, 文件不进入回收站。函数的调用格式如下。
- delete filename: 从磁盘上删除指定的文件, 其中 filename 可以是绝对路径或当前路径下的文件名。
- delete('filename'): 功能同 delete filename 的调用格式, filename 为字符串。
- delete(h): 删除句柄 h 标识的图形对象。

(3) close(): 关闭命令。

close 函数用于关闭指定的图形窗口, 其调用格式如下。

- close: 关闭当前的图形窗口。
- close(h): 删除由句柄 h 标识的图形窗口。
- close all: 删除所有的图形窗口。

(4) reset(): 重置命令。

reset 函数用于重置图形窗口的属性, 使其恢复为默认值。函数的调用格式如下。

- reset(h) 重新设置由句柄 h 标识的图形窗口的属性, 使其恢复到为系统的默认状态。
- reset(gca): 重新设置当前坐标轴对象的属性。
- reset(gcf): 重新设置当前图形窗口的属性。

【例 5.12】图形窗口的常用操作命令的使用。

```

%以下代码用于演示图形窗口常用操作命令
%请读者注意观察图形窗口
h=figure;           %新建图形窗口, 返回句柄 h
x=[0:2*pi];        %在新建图形窗口绘制图形
plot(x,sin(x));
cla                %清除当前图形窗口内的内容, 保留坐标轴
clf               %清除当前图形窗口
close(h)           %关闭句柄 h 标识的图形窗口

```

5.1.3 坐标控制

从前面所绘制的图形来看, 一般 MATLAB 都会自动根据所绘制的图形数据范围, 生成坐标轴, 但是默认状态下的坐标轴范围和精度往往并不能达到用户的要求, 用户需要根据需要设置坐标轴的范围、精度、显示方式等, 这样才能更好地观察图形。MATLAB 提供了对坐标轴的控制函数, 用于根据用户需要编辑完善坐标轴。

1. 坐标轴范围的设置

- axis([Xmin,Xmax,Ymin,Ymax]): 设置坐标轴的范围, 指定当前坐标轴 x 轴和 y 轴的范围, 其中 Xmin 为 x 轴范围下限, Xmax 为 x 轴范围上限, Ymin 为 y 轴范围下限, Ymax 为 y 轴范围上限。
- axis([xmin xmax ymin ymax zmin zmax cmin cmax]): 设置坐标轴 x 轴、y 轴和 z 轴的范围,

以及坐标轴的颜色显示范围。

- `xlim([xmin xmax])`: 仅设置 x 轴范围。
- `ylim([ymin ymax])`: 仅设置 y 轴范围。
- `zlim([zmin zmax])`: 仅设置 z 轴范围。
- `axis tight`: 按紧凑方式显示坐标轴范围, 即坐标轴范围为绘图数据范围。

【例 5.13】 图形窗口的坐标轴范围的设置。

```
x = -pi:pi/100:pi;
plot(x,sin(x))           %绘制示例基本图形
axis([-3.5,3.5,-2,2])    %坐标轴范围设置
xlim([-pi pi])           %仅设置 x 轴范围
```

2. 坐标轴刻度的设置

坐标轴刻度精度的设置是通过坐标轴的刻度属性进行设置, 需要通过坐标轴句柄使用函数 `set` 完成设置。其中坐标轴对象句柄的获取主要通过 `h=gca` 语句返回当前坐标系句柄。

坐标轴刻度范围设置具体的调用格式如下。

- `set(gca,'XTick',[XTickmin:XTickstep:XTickmax])`: 设置数字刻度的显示范围和精度, 与图形做图数据范围相对应。
- `set(gca,'XTickLabel',[XTickLabelmin:XTickLabelstep:XTickLabelmax])`: 设置坐标轴刻度线下的数值显示, 默认状态下为做图数据相应坐标轴数据范围和刻度。
- `set(gca,'XTickLabel',string)`: 设置文本坐标轴刻度。

坐标系横纵坐标轴刻度的比例往往是根据数据自动设置的, 比例有时可能不一样, 通过执行语句 `axis equal` 可以获得等比例的坐标轴刻度。

【例 5.14】 图形窗口的坐标轴刻度的设置。

```
x = -pi:pi/100:pi;
plot(x,sin(x))
set(gca,'XTick',[-pi:pi/2:pi]);           % 设置 x 轴的刻度线
set(gca,'XTickLabel',{'-pi' '-pi/2' '0' 'pi/2' 'pi' }) % 设置 x 轴的文本刻度
axis equal                                 % 等刻度坐标轴的获取
```

执行上述程序, 最后获得的设置了坐标轴刻度的图形如图 5.21 所示。

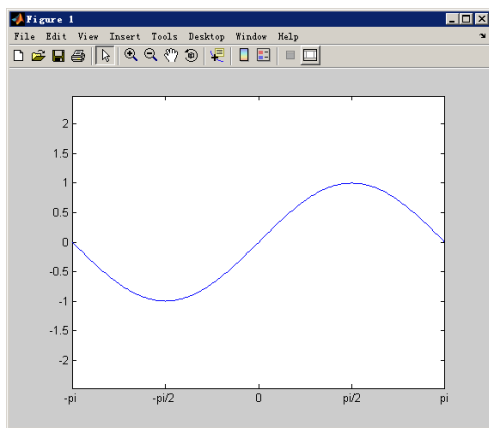


图 5.21 图形窗口的坐标轴刻度的设置

3. 坐标轴字体的设置

坐标轴字体的设置主要通过对字体的属性进行设置, 常用的字体属性如下。

- `FontName` 属性: 字体的类型属性, 包括常用的字体类型。
- `FontSize` 属性: 字体的大小属性。

- FontUnits 属性：字体的单位属性。
- FontWeight 属性：字体样式属性，包括 normal（正常）、bold（加粗）、light（倾斜）、demi（黑体）。

【例 5.15】图形窗口的坐标轴字体设置。

```
x = -pi:pi/100:pi;
plot(x,sin(x));
set(gca,'FontName','Times New Roman','FontSize',16,'FontWeight','Bold');%设置坐标轴的字体类型、大小、样式
```

4. 坐标轴边框的设置

坐标轴边框常用的属性主要有颜色属性，以 X 轴为例，列举坐标轴边框属性。

- XDir 属性：控制 X 轴方向属性，默认状态下属性值为“normal（正常）”，可选属性值有“reverse（逆转）”。
- XColor 属性：设置 X 轴边框的颜色属性。
- LineStyleOrder 属性：设置坐标轴边框的线条类型属性。
- LineWidth 属性：设置坐标轴边框的线条颜色属性。

5. 坐标轴形状和位置的设置

类似于图形窗口大小的设置，坐标轴位置和大小的设置同样可以通过设置坐标轴对象的“Position”属性来完成，其设置的语法格式如下。

```
set(gca,'Position',[left,bottom,width,height])
```

向量 [left,bottom,width,height] 定义坐标轴在图形窗口的位置与大小，默认状态下的单位为 normalized（标准化单位）。

通常情况下，MATLAB 的坐标系是长方形，长宽比大约是 4:3，但是有时可能需要设置正方形的坐标系，可以通过语句 axis square 设置。

【例 5.16】图形窗口的坐标轴形状和位置设置。

```
x = -pi:pi/100:pi;
plot(x,sin(x));
get(gca,'Position'); %返回当前坐标轴对象的位置和大小信息
set(gca,'Position',[0.1,0.2,0.3,0.4]); %设置当前坐标轴对象的位置和大小信息
axis square %设置为相等的坐标轴系统
```

执行上述程序，显示如图 5.22 所示的图形。

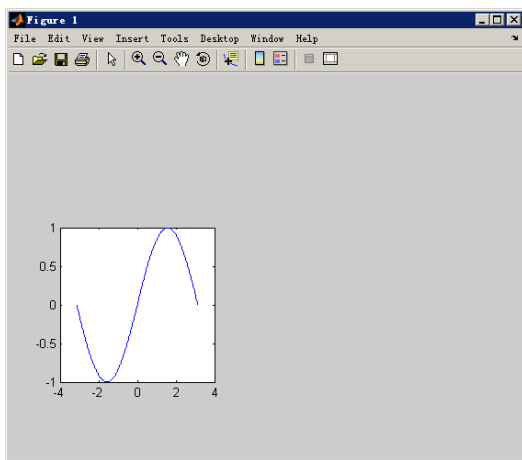


图 5.22 图形窗口的坐标轴形状和位置设置

6. 坐标轴的显示控制

坐标轴显示的控制通过命令 `axis on` 和 `axis off` 控制，默认状态下开启。

`XGrid / YGrid` 属性用于控制坐标轴网格线的显示。

【例 5.17】 图形窗口的坐标轴显示设置。

```
x = -pi:pi/100:pi;
plot(x,sin(x));
set(gca, 'XGrid','on');           %显示 x 坐标轴的网格线
axis off;                          %清除坐标轴
```

5.1.4 图形标注

在完成了一幅图形的绘制工作后，为了让他人更好地理解图形的意义，可以添加图形标注，进一步完善图形，更好地说明图形中数据的含义。MATLAB 7.0 中的图形标注包括图形标题、坐标轴、文本和图例的标注。

1. 标题标注的设置

函数 `title` 用于给当前图形坐标轴的正上方添加标题。其具体的调用格式如下。

- `title('string')`: 在图形窗口添加字符串 `string` 作为标题。
- `title('string','PropertyName',PropertyValue,...)`: 在图形窗口添加标题，并对标题的格式设置。

2. 坐标轴标注的设置

函数 `xlabel` 和 `ylabel` 分别用于 `x` 轴和 `y` 轴的坐标轴标注设置，其用法如下。

- `xlabel('string') / ylabel('string')`: 分别为当前坐标轴对象中 `x` 轴和 `y` 轴添加标注。
- `xlabel('string','PropertyName',PropertyValue, ...) / ylabel('string','PropertyName',PropertyValue)`: 为坐标轴添加标注，同时设置坐标轴对象添加标注的属性。

3. 文本标注的设置

函数 `text()` 和 `gtext()` 可用于文本标注，其中 `text()` 需要设置文本标注的位置，而函数 `gtext()` 用于交互式的文本标注，函数执行后由用户在图形窗口中选择标注的位置。函数的调用格式如下。

- `text(x,y, 'string')`: 函数用于在图形指定位置 `(x,y)` 上标注字符串 `string`，`x`、`y` 为坐标轴实际数值的标注。
- `gtext('string')`: 函数用于交互式的标注在图形上标注字符串 `string`，函数执行后，图形中将出现“十”字型交叉线让用户选择待标注的位置。

4. 图例标注的设置

函数 `legend()` 用于在图形中添加图例标注，图例设置不同的图形颜色、线型等所代表的数据的意义，在 MATLAB 的多种图形中都可以生成图例。为图形添加图例的函数为 `legend()`，其调用格式如下。

- `legend('string1','string2',...)`: 为图形中各部分数据添加图例，字符串 `string1`、`string2`... 按照数据显示的顺序依次标注各部分数据的图例。
- `legend('string1','string2',pos)`: `pos` 参数用于设置添加的图例的位置，默认状态下生成的图例可能把图形部分区域遮盖，`pos` 可以取 -1~4 范围内的整数，按序分别代表：图形窗口右边、图形窗口之内（尽量不与图形遮盖）、图形窗口右上角、图形窗口左上角、图形窗口左下角、图形窗口右下角，同时图例也可以通过鼠标直接移动。

【例 5.18】 图形窗口标注的使用。

```

x = -pi:pi/100:pi;
plot(x,sin(x)); %绘制基本图形
title('图形标注示例','FontName','黑体','FontSize',16) %添加图题标注
xlabel('x=-pi:pi/100:pi','FontName','Times New Roman','FontSize',16) %为 x 轴添加标注
ylabel('三角函数','FontName','黑体','FontSize',16) %为 y 轴添加标注
axis tight %坐标轴以紧凑方式显示
%在 x 轴数值 0.1, y 轴数值 0.3 的位置添加文本注释, 并设置文本的字型和字号
text(0.1,0.3, 'y=sin(x)','FontName','Times New Roman','FontSize',16)
hold on; %图形保持
plot(x,cos(x),'ro'); %在原图形上, 添加新的图形
gtext('y=cos(x)','FontName','Times New Roman','FontSize',16) %交互式的添加文本
legend('正弦函数','余弦函数',-1) %添加图例, 显示在图形的右边

```

执行上述程序, 将生成如图 5.23 所示的图形。

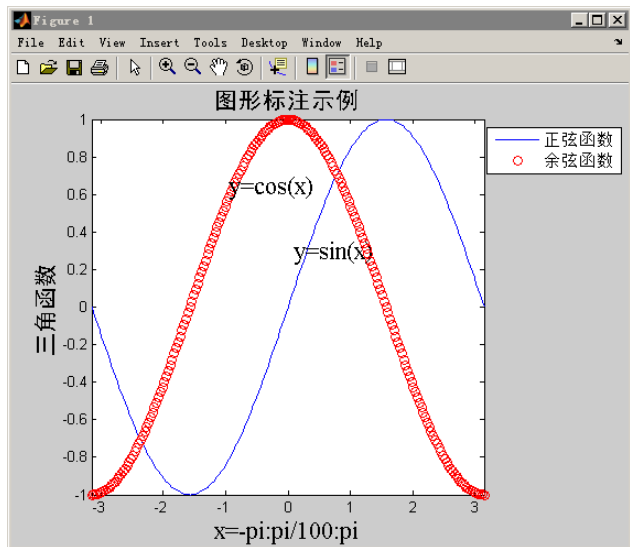


图 5.23 图形标注的使用

5.1.5 窗口分割

函数 `subplot()` 用于图形窗口的分割, 即在同一个图形窗口可以同时显示多个坐标轴的图形。此函数可以用于设置多图形的同时显示, 便于观察比对。`subplot()` 函数的使用原理为首先把图形窗口分为多个区域, 然后依次在各区域绘制图形, 其调用格式如下。

`subplot(m,n,p)`: 函数把图形窗口分为 $m \times n$ 个绘图子区, 在第 p 个绘图子区绘制图形, 绘图子区的编号按行方向编号。

【例 5.19】图形窗口分割的使用。

```

%图形窗口分割的使用, 并依次在相应的窗口绘制图形
subplot(2,3,1);
ezplot('sin');
subplot(2,3,2);
ezplot('cos');
subplot(2,3,3);
ezplot('tan');
subplot(2,3,4);
ezplot('cot');
subplot(2,3,5);
ezplot('sec');
subplot(2,3,6);
ezplot('csc');

```


执行上述程序，显示如图 5.24 所示的图形。

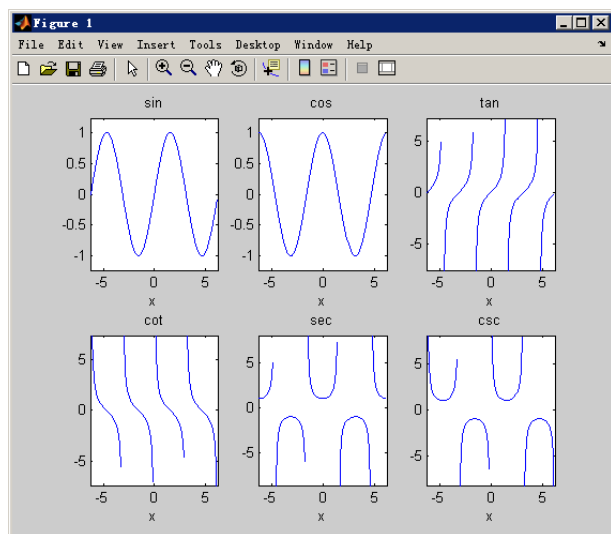


图 5.24 图形窗口分割的使用

5.1.6 MATLAB 图形编辑工具的使用

MATLAB 图形编辑除了前面章节中介绍的通过相关命令、函数设置外，在绘制图形后也可以直接在图形窗口中进行图形的编辑。如果所绘制的图形不需要重复绘制，相比而言在图形窗口直接编辑更为直观、方便，编辑完直接可以查看图形编辑的效果，且不用查阅相关的函数命令，直接单击相应的设置区域，图形窗口会有相关属性设置的选项，直接通过鼠标即可完成操作，对于入门读者来说更为简单、方便。

利用界面操作进行 MATLAB 图形编辑主要在图形窗口实现，下面将以 MATLAB 图形窗口的介绍为基础，向读者讲述 MATLAB 图形编辑工具的使用。

1. 图形窗口的菜单栏

MATLAB 图形窗口的菜单栏主要包括“File”、“Edit”、“View”、“Insert”、“Tools”、“Desktop”、“Windows”和“Help”。其中部分菜单项与 MATLAB 主界面的菜单项类似，这里主要介绍菜单项中图形窗口所特有的功能。

1) “File”子菜单

“File”子菜单中重点要讲述的是图形的保存输出功能，如图 5.25 所示，涉及的菜单项包括如下几个。

- “Save”菜单项：用于图形的保存功能，可以保存为常用的多种图片格式。
- “Save As”菜单项：用于图形的另存为功能，可以保存为常用的多种图片格式。
- “Generate M-File”菜单项：生成创建图形的 MATLAB 代码，代码中保存当前创建的图形对象设置的属性，下次可以利用代码创建相同的图形。
- “Export Setup”菜单项：用于图形输出设置。选择“File”→“Export Setup”命令将打开如图 5.26 所示的“Export Setup”窗口。在此窗口中，包含输出 Properties（属性）和 Export Styles（样式）的设置。其中 Properties 设置中又包含了 Size、Rendering、Fonts、Lines 属性的设置，Size 属性用于设置图形的大小，Rendering 属性用于设置颜色模式和分辨率，其中对于有较高打印要求的图像需要提高分辨率，Fonts 属性用于设置图形字体类型和字

号, Lines 属性用于对图形中线条的设置。设置完毕后, 可以单击“Export Setup”窗口中的“Export”按钮保存设置的图形属性, 以备下次重复使用。

- “Page Setup”菜单项: 用于页面设置。选择“File”→“Page Setup”命令将打开如图 5.27 所示的“Page Setup”对话框。在此对话框中, 包含 4 个选项卡: Size and Position (图形尺寸和位置)、Paper (纸张)、Lines and Text (线型和文本) 和 Axes and Figure (坐标轴和图形)。
- “Print Setup”菜单项: 用于打印设置。
- “Print Preview”菜单项: 用于打印预览。
- “Print”菜单项: 用于直接打印图形。

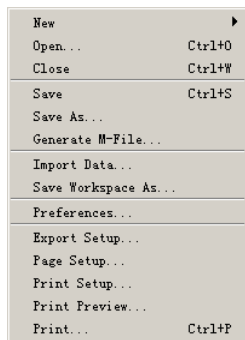


图 5.25 Figure 窗口“File”子菜单

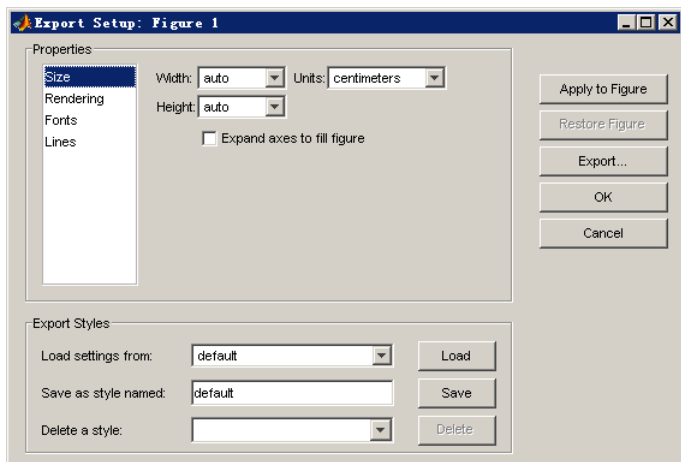


图 5.26 “Export Setup”窗口

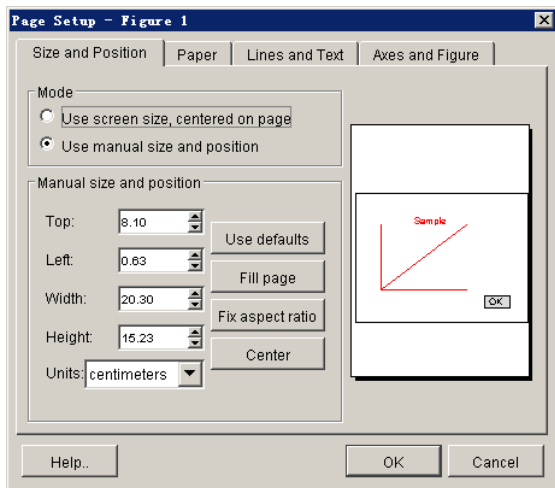


图 5.27 “Page Setup”对话框

2) “Edit”子菜单

“Edit”子菜单主要用于对图形的编辑, 常用的菜单项如下。

- “Copy Figure”菜单项: 复制当前的图形对象, 并且直接粘贴到 Word 等文件中, 保存图片, 且图片的清晰度较高。
- “Copy Options”菜单项: 图形复制时参数的设置。选择“Edit”→“Copy Options”命令, 如

图 5.28 所示，将打开如图 5.29 所示的复制属性设置页面。在此窗口中包括：Clipboard format（复制形式）设置，Figure background color（图片背景色）设置，Size（大小）设置。

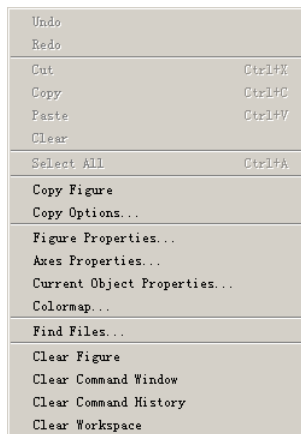


图 5.28 Figure 窗口“Edit”子菜单

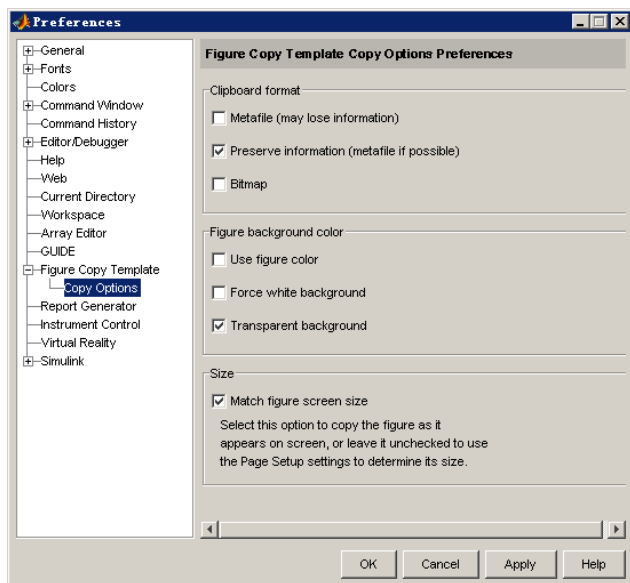


图 5.29 “Copy Options”设置窗口

- “Figure Properties”菜单项：打开图形窗口属性设置对话框，可以设置图形颜色、标题、显示类型等属性。选择“Edit”→“Figure Properties”命令在原图形下方将打开图形窗口设置对话框，如图 5.30 所示。在界面中显示的可以设置的属性有 Figure Name（图形名称），Colormap（色图控制），Figure Color（颜色）和 Show Figure Nameber（是否显示图形名称）。其他的一些属性，可以通过单击“Figure Properties”对话框中的“Inspector”按钮，打开如图 5.31 所示的图形窗口属性查看器，查看并修改设置各属性。设置完毕可以通过“Export Setup”按钮保存设置的图形属性。

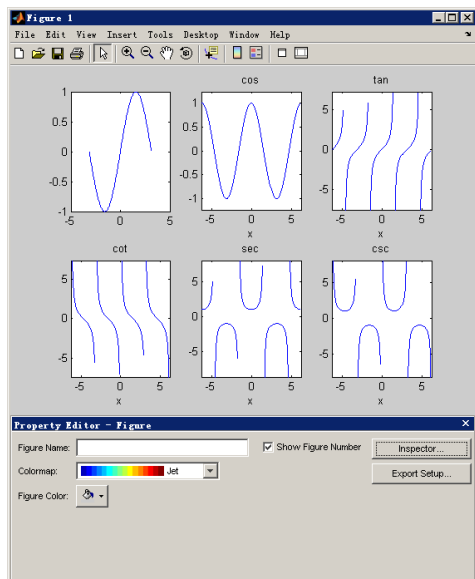


图 5.30 “Property Editor”设置对话框

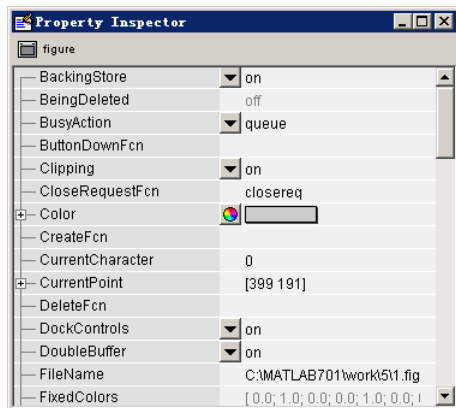


图 5.31 “Property Inspector”窗口

- “Axes Properties” 菜单项：打开图形窗口坐标轴对象的属性设置对话框，可以设置坐标范围、刻度、比例、标注等信息。选择 “Edit” → “Axes Properties” 命令，在原图形下方将打开坐标轴设置对话框，如图 5.32 所示。在界面中包含 x 轴、y 轴、z 轴设置页面，打开相应的选项，对不同坐标轴设置。对每一坐标轴，界面中显示的可以设置的属性有坐标轴的标注、标轴的数据范围、坐标轴的尺度、字体等，其中对于刻度及其刻度下的显示需要单击 “Ticks” 按钮，打开如图 5.33 所示的刻度设置对话框设置。刻度的设置可以选择 Auto (自动)，Manual (人工)，Step by (按指定步长)，同时可以手动直接在中下方的表格中输入相应的刻度。同时其他一些坐标轴的属性也可以通过单击 “Inspector” 按钮，打开坐标轴属性查看器，设置属性值。

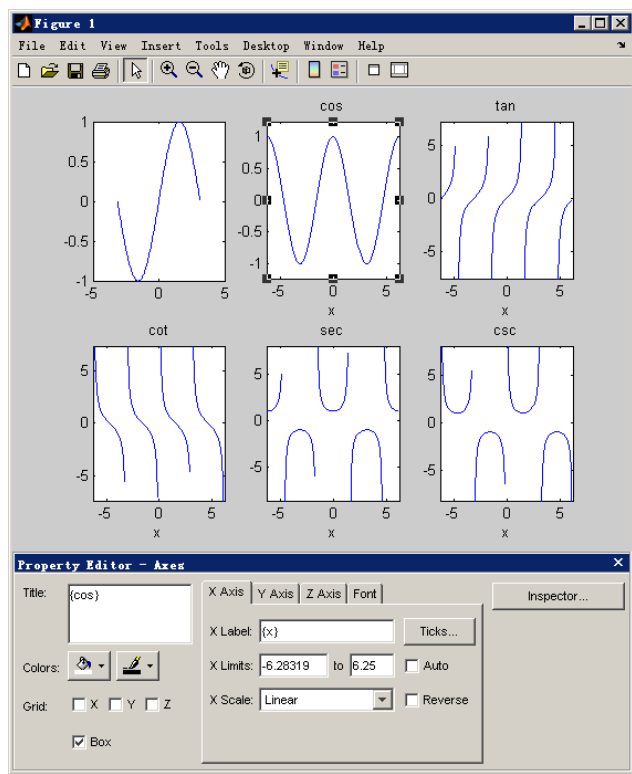



图 5.32 “Property Editor” 设置对话框

- “Current Object Properties” 菜单项：打开当前对象的属性设置对话框，通过标题栏中的图标  选择当前对象，再单击此菜单，将打开当前对象的属性设置对话框；此菜单在属性设置中是很有用的，需要设置什么的属性，直接用鼠标选中即可，所见即所得，与之前的属性查看器设置其他属性相比，对于入门读者来说更为方便、实用。
- “Colormap” 菜单项：用于设置色图的模式，色图是指以不同的颜色对应不同的数值，其中的对应方式即为 Colormap 菜单项设置的内容。选择 “Edit” → “Colormap” 命令，将打开如图 5.34 所示的 “Colormap Editor” 窗口，在此窗口中，单击 Tools 菜单，可以选择不同的色图模式，有 “autumn”，“blue” 等，颜色映射中数据的范围通过 “color data min” 和 “color data max” 文本框设置。

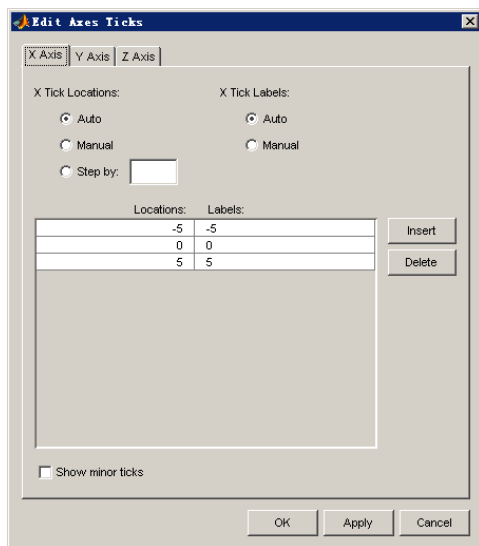


图 5.33 “Edit Axes Ticks” 设置对话框

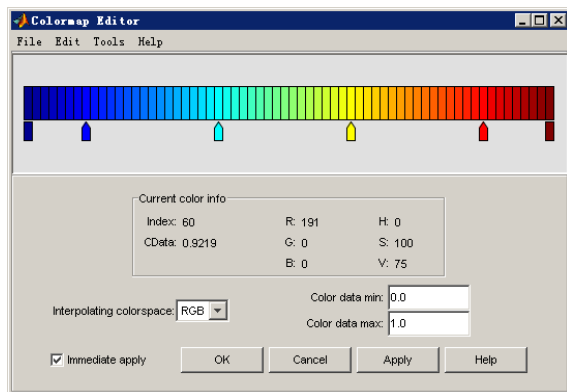


图 5.34 “Colormap Editor” 窗口

3) “View” 子菜单

“View” 子菜单用于决定不同工具条和对话框的显示，如图 5.35 所示，单击菜单项，即显示相应的工具条，“View” 子菜单中在图形窗口显示的工具条前将有“√”。

- “Figure Toolbar” 菜单项：控制图形窗口中工具栏的显示。
- “Camera Toolbar” 菜单项：控制图形窗口中照相操作工具栏的显示。
- “Plot Edit Toolbar” 菜单项：控制画图编辑工具条的显示。
- “Figure Palette” 菜单项：控制图画板的显示。
- “Plot Browser” 菜单项：控制绘图浏览器的显示。
- “Property Editor” 菜单项：控制属性编辑器的显示。

当“View”子菜单中不同工具条和对话框都设置为显示状态时，如图 5.36 所示。

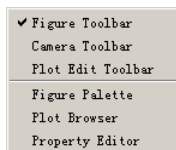


图 5.35 Figure 窗口“View”子菜单

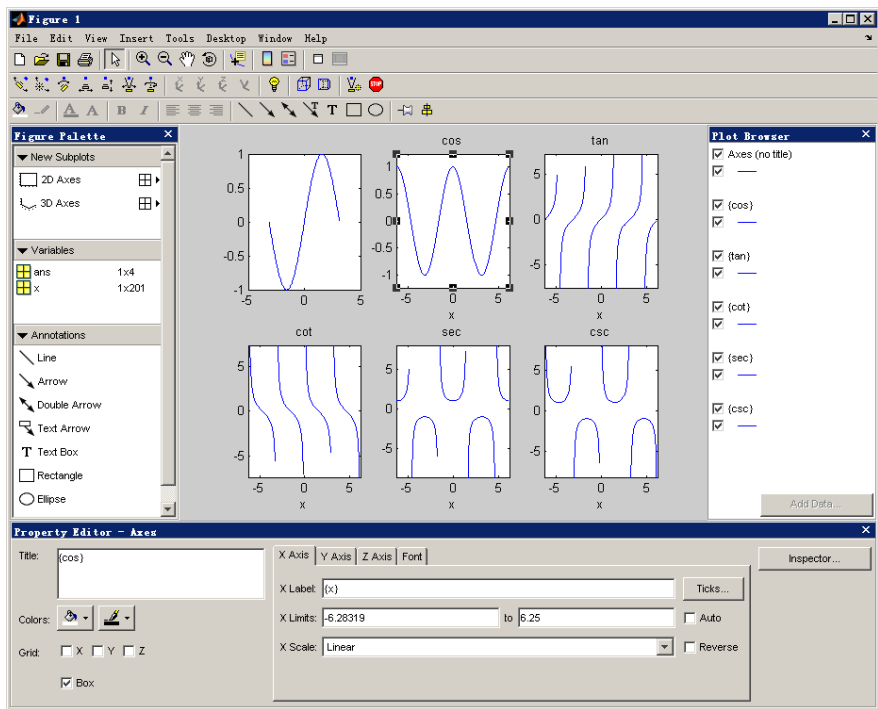


图 5.36 各工具条的显示

4) “Insert” 子菜单

通过“Insert”子菜单可以向图形中添加不同的绘图对象，可以在图形窗口中添加的对象有 X_Label (X 轴)、Y_Label (Y 轴)、Z_Label (Z 轴)、Title (图例)、Legend (图例)、Colorbar (颜色条)、Line (直线)、Arrow (箭头)、Text Arrow (带箭头的文本框)、Double Arrow (双向箭头)、TextBox (文本框)、Rectangle (矩形)、Ellipse (椭圆)、Axes (坐标轴) 和 Light (光源)，如图 5-37 所示。同时 Plot Edit Toolbar (图形编辑工具条)，如图 5.38 所示，也提供了上述这些插入绘图对象的功能。

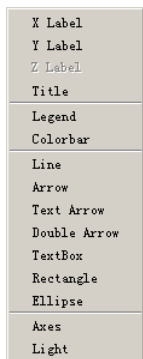


图 5.37 Figure 窗口“Insert”子菜单



图 5.38 Plot Edit Toolbar 工具条

5) “Tools” 子菜单

“Tools”子菜单主要用于提供一些图形编辑的工具，便于更好地观察、编辑图形，如图 5.39 所示。

- “Edit Plot” 菜单项：控制图形编辑状态，当选择该菜单，菜单前有“√”，表示当前图形窗口处于被编辑状态。

- “Zoom In” 菜单项：控制图形放大。
- “Zoom Out” 菜单项：控制图形缩小。
- “Pan” 菜单项：控制手动移动图形。
- “Rotate 3D” 菜单项：控制三维旋转图形，以便从不同角度观察图形。
- “Data Cursor” 菜单项：从图形中显示数据点的坐标。
- “Reset View” 菜单项：重置编辑过的图形。
- “Options” 菜单项：用于上述菜单项的一些附加参数设置，包括缩放设置：Unconstrained Zoom（无限制的缩放）、Horizonted Zoom（水平方向缩放）、Vertical Zoom（垂直方向缩放）；图形移动的控制：Unconstrained Pan（无限制的移动）、Horizonted Pan（水平方向移动）、Vertical Pan（垂直方向移动）；Display Cursor as Datatip（在图形窗口内显示数据点坐标）、Display Cursor in Window（在另外的窗口显示数据点的坐标）。
- “View Layout Grid” 菜单项：控制图形窗口背景网格的显示。
- “Align Distribute Tool” 菜单项：控制图形窗口子窗口的排列布局，单击此菜单项将弹出如图 5.40 所示的“Align Distribute Tool”窗口，设置子窗口的对齐方式。紧接此菜单项的“Align”与“Distribute”菜单项功能与其类似，但是“Align Distribute Tool”窗口在设置子窗口的排列布局方式时更为直观。

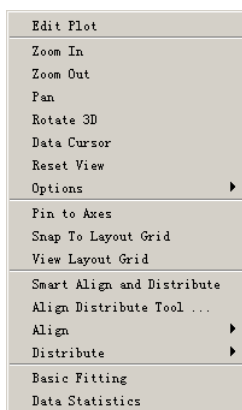


图 5.39 Figure 窗口“Tools”子菜单

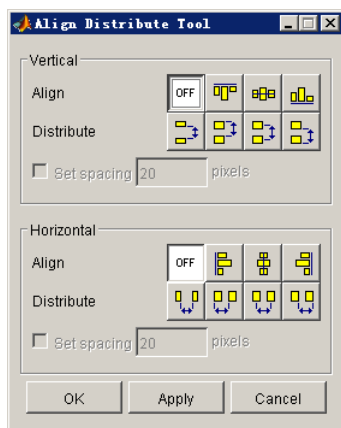


图 5.40 “Align Distribute Tool”窗口

- “Basic Fitting” 菜单项：数据曲线拟合，单击此菜单将打开如图 5.41 所示的“Basic Fitting”窗口。在该窗口中，用户需要 Select data（选择待拟合的数据源）、Center and scale X data（控制数据归一化）、Check to display fits on figure（设置拟合的模型）、Show equations（设置拟合函数的显示）、Significant digits（设置数值的有效位数）、Plot residuals（控制拟合模型残差的绘制）和 Show norm of residuals（控制最大残差模的显示）等。
- “Data Statistics” 菜单项：对绘图数据进行简单地统计分析，单击此菜单将打开如图 5.42 所示的“Data Statistics”窗口。在此窗口中，用户可以获取的统计参数有 min（最小值）、max（最大值）、mean（平均值）、median（中位数）、std（方差）及 range（极差）。同时单击“Save to workspace”按钮，可以把这些统计参数保存至 MATLAB 工作空间中。

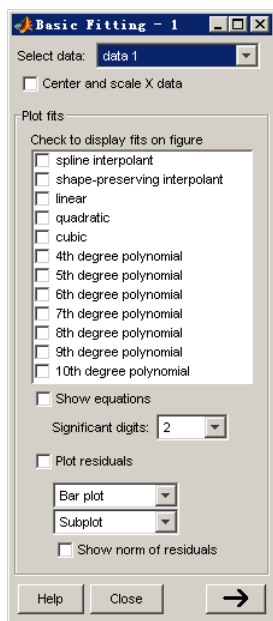


图 5.41 “Basic Fitting” 窗口

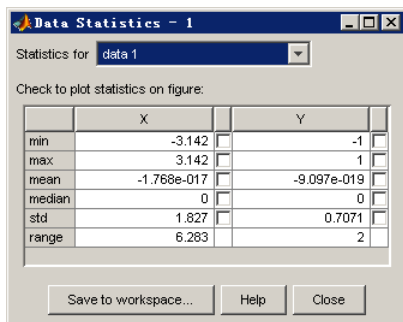


图 5.42 “Data Statistics” 窗口

另外，图形窗口的“Desktop”菜单、“Window”菜单和“Help”菜单，与其他界面中的大致一样，读者可以参照使用，在此不再详细叙述。

2. 图形窗口的工具栏

MATLAB 图形窗口的工具栏位于菜单栏的下方，提供了常用图形编辑的快捷操作图标，主要有：

- 图标 ：用于新建图形窗口。
- 图标 ：打开已有的 fig 格式的图形。
- 图标 ：用于保存图形。
- 图标 ：用于打印图形。
- 图标 ：选定当前对象。
- 图标 ：用于图像的放大，功能同 zoom on 函数。
- 图标 ：用于图像的缩小，功能同 zoom off 函数。
- 图标 ：移动选定的对象，当图形放大后，在图形窗口中可能只能显示部分的图形，通过图形移动工具可以在图形窗口看到不同部分的图形。
- 图标 ：对当前图形进行三维旋转，手动控制旋转方向和角度。
- 图标 ：用于读取图形中的数据点，可以显示图形中任意一点的坐标。
- 图标 ：显示图形的颜色标注，对于不同颜色表示不同数据的图形，需要添加。
- 图标 ：快速添加图例的显示。
- 图标 ：图形窗口布局的设置。

3. 数据浏览窗口的图形绘制

在 MATLAB 工作空间的数据浏览窗口提供了对存储数据的快速绘图方式，如图 5.43 所示，选中数据，单击“workspace”的绘图工具栏，即可快速绘制不同类型的图形，便于数据的观察。

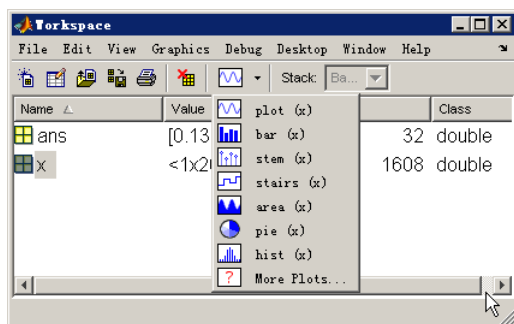


图 5.43 “Workspace” 窗口

5.2 特殊二维图形

在上一节中介绍的二维图形主要是常规的曲线图，而在实际中为了更好地反映数据的变化规律，往往需要绘制一些特殊的二维图形。MATLAB 为用户提供的特殊二维图形包括条形图、面积图、饼图、散点图、柱状图、罗盘图、羽毛图、矢量图、杆型图、阶梯图、极坐标图、等势图等。本小节将重点讲述这部分内容，同时，5.1.6 小节中关于图形编辑工具的使用方法也同样适用于这些二维图形。

5.2.1 条形图

条形图是用单位长度表示一定的数量，各数据变量按照数量的多少画成长短不同的条形，便于比较分析。

二维条形图按图形方向可以分为垂直条形图和水平条形图，而每种图形又都存在两种模式：累计式和分组式。MATLAB 中绘制条形图的基本函数为 `bar()` 和 `barh()`，函数 `bar()` 用于绘制垂直条形图，函数 `barh()` 用于绘制水平条形图。函数 `bar()` 与 `barh()` 绘制条形图时调用格式基本一致，现以 `bar()` 函数为例，介绍条形图函数的调用格式。

- `bar(y)`: 若 `y` 为向量，条形图的高度代表向量 `y` 中每个数据的大小，横坐标即为 1 到 `length(y)`；若 `y` 为矩阵，则把矩阵 `y` 分解成行向量，横坐标为 1 到行数，不同的列以不同的条形系列标识，即如果 `y` 为 $n \times m$ 的矩阵，将把 `y` 矩阵分为 `n` 组，每组 `m` 个列数据的条形图。
- `bar(x,y)`: 以 `x` 为横坐标，绘制数据 `y` 的条形图，其中 `x` 必须为单调递增的向量。
- `bar(y,width)`: 其中参数 `width` 用于设置条形的宽度，默认值为 0.8，若 `width` 设置为 1，则条形之间没有间隔，若 `width` 值大于 1，则条形之间将重合。
- `bar(y,'style')`: `style` 参数用于设置条形图的排列类型，默认情况下为 “group” 类型，按分组式的方法显示条形图，分组式即当 `y` 为时，条形图分为 `n` 组，每组有 `m` 个垂直条形；类型 “stack” 按照累计式方法显示条形图，累计式即对于 $n \times m$ 的矩阵 `y`，每一行向量显示在一个条形中，条形的高度为该行数据的总和，每一列在各条形中用不同的颜色标识。
- `bar(y,LineStyle)`: `LineStyle` 参数用于设置条形图的颜色。

另外，MATLAB 也提供了函数 `bar3()` 与 `bar3h()` 分别用于绘制水平和垂直的三维条形图，其调用格式同函数 `bar()` 和 `barh()`，在此不再详细叙述，感兴趣的读者可以阅读相关的帮助文档。

【例 5.20】 二维条形图的绘制。

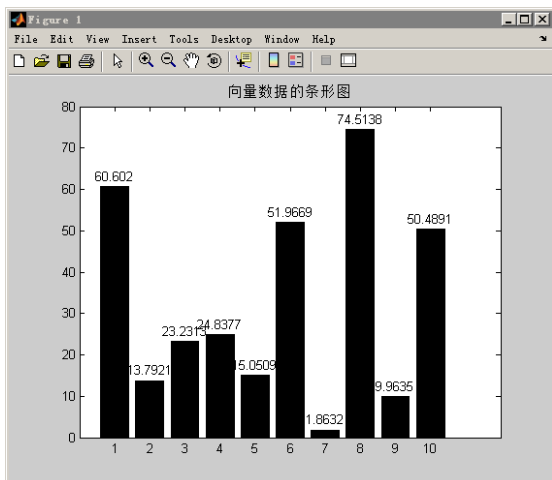
```
y=abs(rand(1,10)*100); %生成随机向量 y
bar(y)                  %绘制条形图
```

```

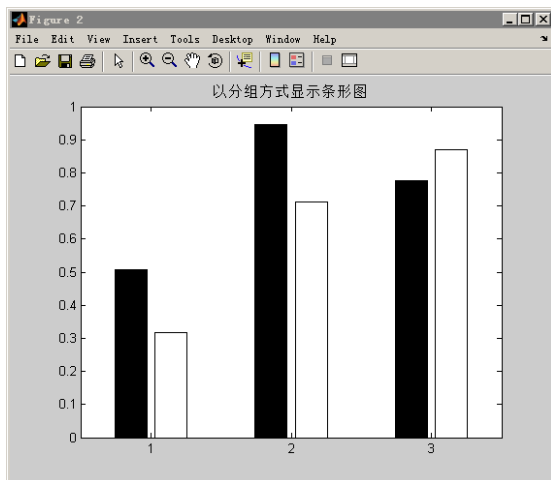
for i=1:10                                %在条形图上标注数值
text(i,y(i)+0.5,num2str(y(i)),'VerticalAlignment','bottom','HorizontalAlignment','center');
end
title('向量数据的条形图');
x=1:3;
y=rand(3,2);
figure;
bar(x,y);                                %生成矩阵 y 的条形图，规定横坐标数据，并以分组方式显示条形图
title('以分组方式显示条形图');
figure;
bar(x,y,'stack');                        %生成矩阵 y 的条形图，按累计方式显示条形图
title('以累计方式显示条形图');
width=1;
figure;
bar(x,y,width);                          %设置条形图宽度为“1”
title('条形图的宽度为1');

```

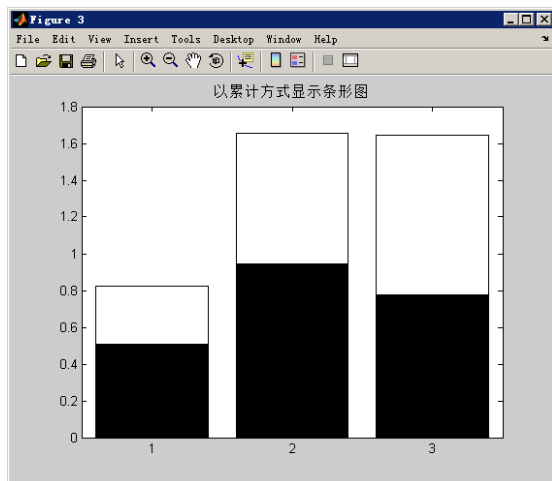
执行上述程序，将生成如图 5.44 所示的各条形图。



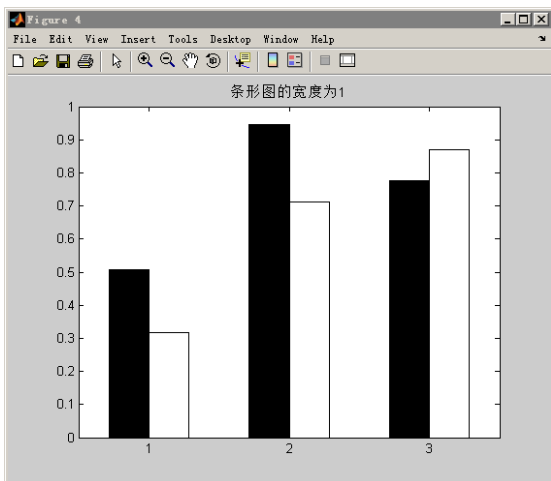
(a)



(b)



(c)



(d)

图 5.44 二维条形图的绘制

5.2.2 直方图

直方图是根据数据的分布情况,对数据进行分组,以组距为底边、以频数为高度的连接起来的直方型矩形图。直方图和条形图的形状类似,但其区别很大:(1)直方图是用矩形的面积表示各组数据的多少,矩形的高度表示每一组的频数或频率,宽度则表示各组的组距,其高度与宽度均有实际意义。条形图是用条形的长度表示各数据的多少,其宽度则是固定的,可任意设置,无实际意义;(2)直方图的各矩形由于分组数据的连续性,一般情况下是连续排列的,而条形图在条形没有设置过大的情况下是分开排列的。

MATLAB 提供的绘制直方图的函数为 `hist()`,其调用格式如下。

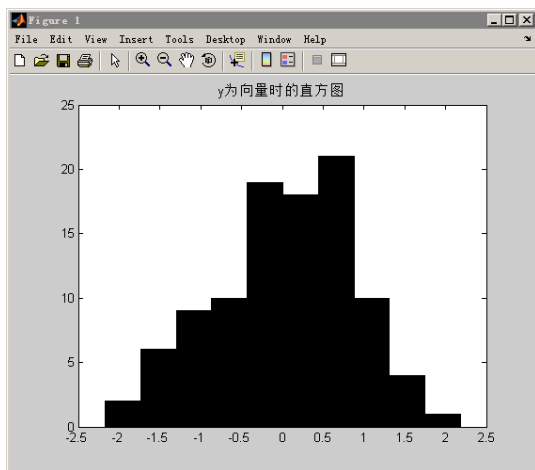
- `n=hist(y)`: 根据向量 `y` 中元素的大小分布,向量 `y` 的数据被划分为 10 个分组,绘制各个分组中元素个数的情况,且返回每一个分组中的元素个数。若 `y` 为矩阵,则函数按列对矩阵 `y` 进行处理。
- `n=hist(y,x)`: 输入参数 `x` 为向量,则以 `x` 值为直方图条形的中心位置,直方图条形的数目即为 `length(x)`。
- `n=hist(y,m)`: 输入参数 `m` 为标量,用于指定直方图条形的数目。

以上各输入格式如果没有参数 `n` 返回,即绘制直方图,否则仅返回各分组中的元素情况。

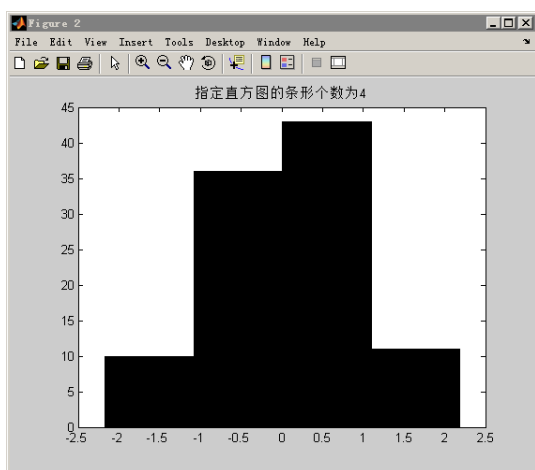
【例 5.21】直方图的绘制。

```
%%
y=randn(100,1);
hist(y);                %y 为向量时直方图的绘制
title('y 为向量时的直方图');
%%
n=hist(y);
disp('直方图 1 的数据分组情况 (每组中数据个数):')
disp(n);
%%
figure
hist(y,4);              %指定条形个数的直方图的绘制
title('指定直方图的条形个数为 4');
%%
figure
x=0.1:0.2:1
hist(y,x);              %指定条形个数的直方图的绘制
title('指定分组的组中心为 0.1、0.3、0.5、0.7、0.9 绘图');
%%
figure;
y=rand(100,3);          %y 为矩阵的直方图
hist(y)
title('y 为矩阵时的直方图');
```

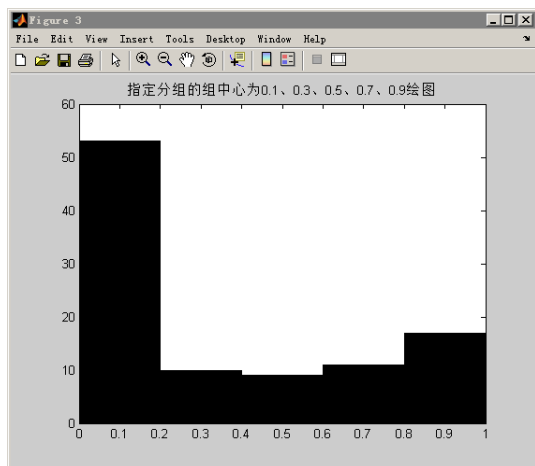
执行上述程序,显示如图 5.45 所示的各图形。



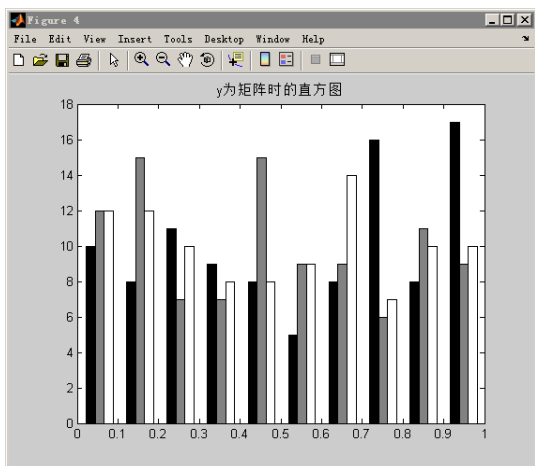
(a)



(b)



(c)



(d)

图 5.45 直方图的绘制

另外，函数 `rose()` 可在极坐标内绘制直方图，不是很常用，感兴趣的读者可以查阅 MATLAB 的相关帮助文档。

5.2.3 面积图

面积图将数据点显示为一组由线连接的点，并填充线下方的所有区域。MATLAB 中绘制面积图的函数为 `area()`，其调用格式如下。

- `area(y)`: 如果 y 为向量，则以 y 的下标为横坐标， y 中各数据的点连接为线，默认情况下以 x 轴为基准线，填充 x 轴与数据点连成的线间的区域。如果 y 为矩阵，则对矩阵的各列操作，绘制多条曲线，每列数据都是以前列数据绘制的曲线为基线，在此基础上绘制而成，并填充不同颜色。
- `area(x,y)`: 指定绘制的面积图的横坐标。
- `area(...,basevalue)`: 设置面积图绘制的基线，`basevalue` 为标量，默认情况下 `basevalue` 值为 0，即以 x 轴为基线，填充第一条曲线的面积区域。

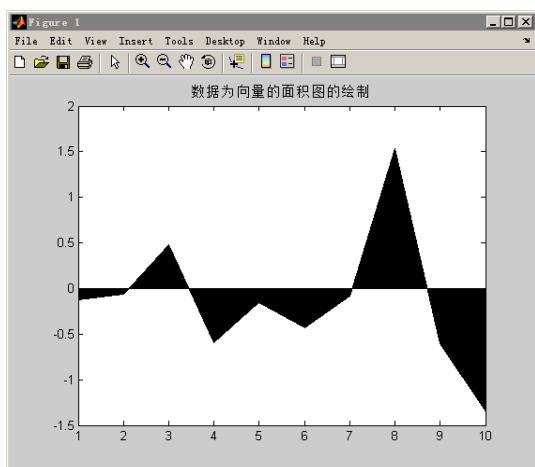
【例 5.22】 面积图的绘制。

```

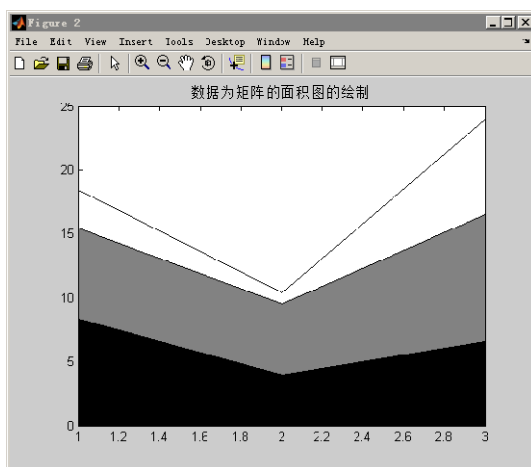
close all;
%%
y=randn(1,10);
area(y);
title('数据为向量的面积图的绘制');
%%
figure;
y=abs(rand(3,3)*10);
area(y);
title('数据为矩阵的面积图的绘制');
%%
figure;
x=1:3:7;
area(x,y);
title('指定横坐标的面积图的绘制');
%%
figure;
x=1:3:7;
area(x,y,2);
title('指定基线为3的面积图的绘制');

```

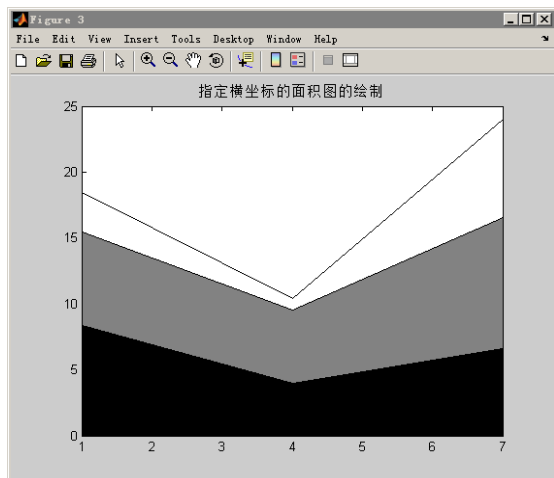
执行上述程序，生成如图 5.46 所示的面积图。



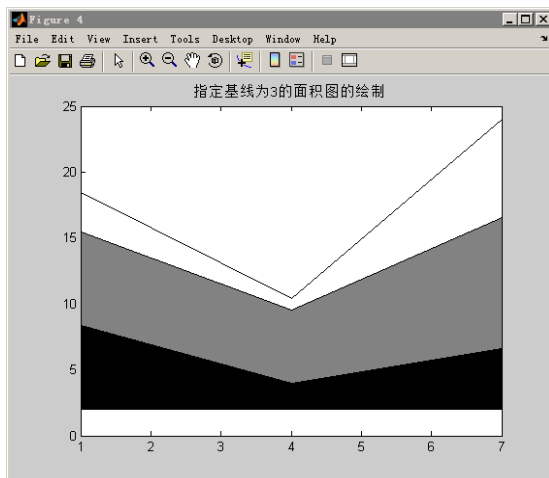
(a)



(b)



(c)



(d)

图 5.46 面积图的绘制

5.2.4 饼图

饼图是一个被划分为多个扇区的圆形图表，每个扇区代表一个数据项，描述各数据项占数据总和的比例。MATLAB 中提供了函数 `pie()` 绘制二维的饼图，其调用格式如下。

- `pie(x)`: 对数据 `x` 绘制饼图，`x` 可为向量或矩阵，`x` 中的每一元素将代表饼图的一扇区，同时饼图中显示各元素在元素总和的比例。
- `pie(x,explode)`: 参数 `explode` 用于设置饼图中抽取式显示的部分，`explode` 向量的长度与 `x` 中元素个数相等，并与 `x` 中元素一一对应，`explode` 元素为非零值，对应的元素扇区将从饼图中分离出来。
- `pie(x,explode,labels)/pie(x,labels)`: 函数输入参数中 `labels` 用于标注各扇区的标注，参数 `labels` 为字符串。

另外，MATLAB 也提供了函数 `pie3()` 用于绘制三维饼图，函数的调用格式与 `pie()` 类似，感兴趣的读者可以查询相关的帮助文档。

【例 5.23】二维饼图的绘制。

```
close all;
%%绘制饼图，分析某考生考试成绩中各考试题型的得分情况
x=[10 30 20 25];           %考试总成绩中各部分的组成
pie(x);                     %绘制饼图，显示各部分的百分比
figure;
pie(x,{'选择','填空','简答','问答'}); %饼图各部分添加标注
执行上述程序，显示如图 5.47 所示的图形。
```

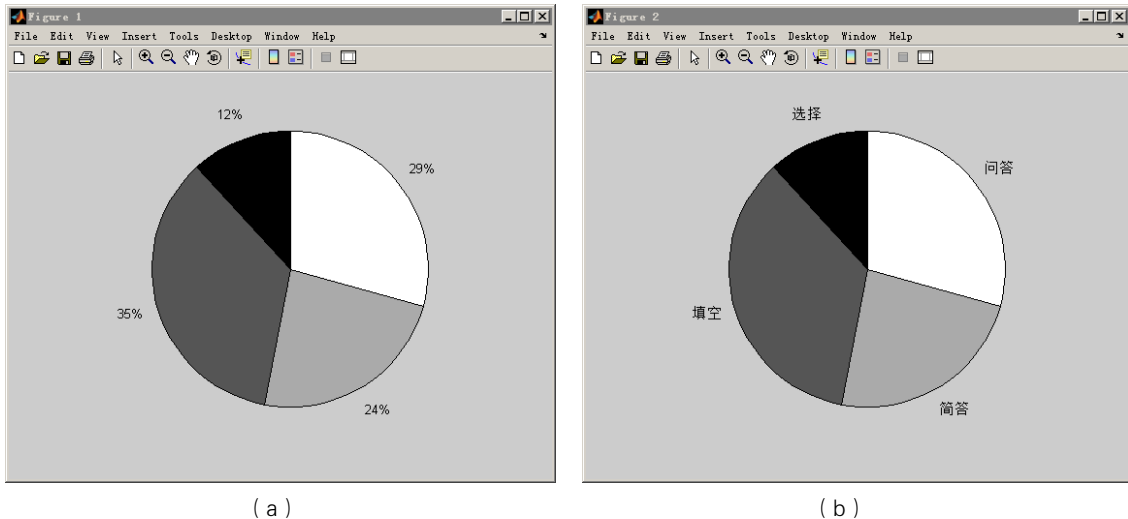


图 5.47 二维饼图的绘制

5.2.5 散点图

散点图将数据序列显示为一组点。在回归分析中较为常用，反映了因变量随自变量而变化的趋势，便于观察两者关系。MATLAB 中提供的绘制散点图的函数为 `scatter()`，其调用格式如下。

- `scatter(x,y)`: 以 `x` 中元素为横坐标，`y` 中元素为纵坐标，绘制二维散点图。
- `scatter(x,y,s)`: 参数 `s` 设置散点的大小。
- `scatter(x,y,s,c)`: 参数 `c` 设置散点的颜色。

【例 5.24】散点图的绘制。

```

x=1:2:20;
scatter(x,2*x);                %绘制散点图
hold on;
scatter(x,3*x,10);             %设置散点的大小
scatter(x,4*x,10,'r');         %设置散点的颜色

```

执行上述程序，显示如图 5.48 所示的图形。

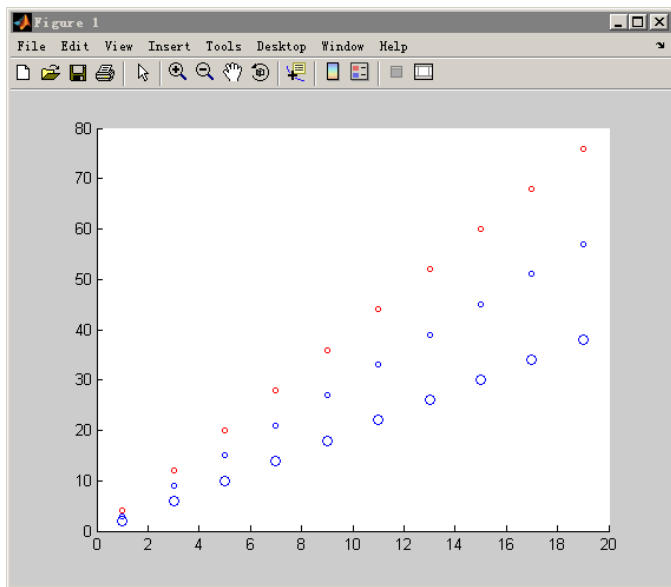


图 5.48 散点图的绘制

5.2.6 排列图

排列图又称累托 (Pareto) 图，用于寻找主要问题或主要原因所使用的图。它是由两个纵坐标、一个横坐标、几个按高低顺序依次排列的条形和一条累计百分比的折线组成。其中，左纵坐标表示频数，右纵坐标表示频率，横坐标表示各因素，按各因素高低从左到右显示，折线表示累积的频率。通过排列图可以较好的分析各因素的重要性。函数 `pareto()` 用于绘制排列图，其调用格式如下。

- `pareto(y)`: 绘制数据 y 的排列图，即排列图条形的高度代表数据 y 值的大小，图的横坐标为各元素在数据 y 中的下标。
- `pareto(y,names)`: 绘制数据 y 的排列图，指定字符串型的横坐标。
- `pareto(y,x)`: 绘制数据 y 的排列图，指定数值型的横坐标。

【例 5.25】排列图的绘制。

```

close all;
y=rand(1,5);
pareto(y);
title('排列图绘制');
figure;
pareto(y,{'a','b','c','d','e'});
title('指定横坐标的排列图绘制');

```

执行上述程序，显示如图 5.49 所示的图形。

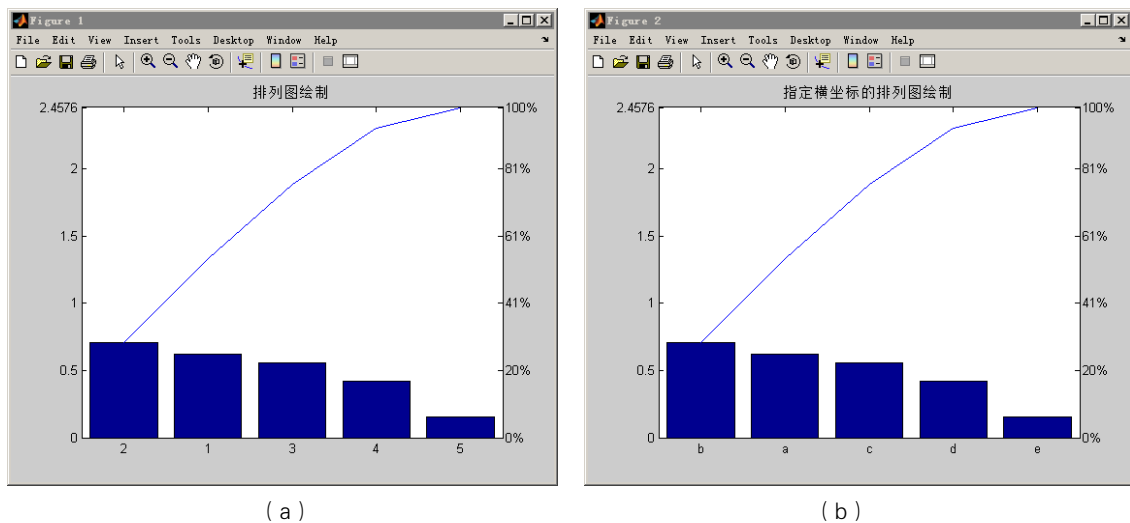


图 5.49 排列图的绘制

5.2.7 罗盘图

MATLAB 中提供了函数 `compass()` 绘制罗盘图，罗盘图绘制于一个圆盘中，从原点出发的箭头，箭头在圆盘中的角度用于表示数据的角度，箭头的长短用于表示数据的大小。函数 `compass()` 的调用格式如下：

- `compass(u,v)`：输入参数 u 、 v 分别指定数据在罗盘图中的 x 分量和 y 分量。
- `compass(z)`：绘制仅一个输入参数的罗盘图， z 为复数矩阵，复数的实部代表罗盘图的 x 分量，虚部代表 y 分量。
- `compass(...,LineSpec)`：绘制罗盘图，设置罗盘图的线型。

【例 5.26】罗盘图的绘制。

```
%% 绘制 12 小时的风向和风力变化图
winddir=[30 90 45 60 180 330 215 60 75 250 300 270]; %设置风向，为角度数据
wind=2:2:24; %设置各个方向的风力
winddir2=winddir*pi/180; %风向数据由角度转换为弧度
[u,v]=pol2cart(winddir2,wind); %将风向和风力的极坐标转换为直角坐标
compass(u,v); %绘制罗盘图
title('罗盘图的绘制');
```

执行上述程序，显示如图 5.50 所示的图形。

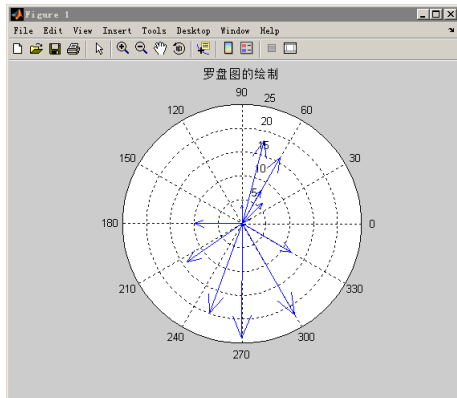


图 5.50 罗盘图的绘制

5.2.8 羽毛图

羽毛图是以箭头的形式绘制矢量数据，与罗盘图不同的是数据绘制于直角坐标系中。MATLAB 中绘制羽毛图的函数为 `feather()`，其调用格式如下。

- `feather(u,v)`: 输入参数 `u`、`v` 分别指定数据在羽毛图中对应的 `x` 分量和 `y` 分量。
- `feather(z)`: 输入参数 `z` 为复数矩阵，复数的实部代表羽毛图的 `x` 分量，虚部代表 `y` 分量。
- `feather(...,LineStyle)`: 指定羽毛图绘制的线性。

【例 5.27】 羽毛图的绘制。

```
%% 羽毛图的绘制
theta = (-90:10:90)*pi/180;           %设置羽毛图的方向，为弧度制
r = 2*ones(size(theta));               %设置各个方向的矢量长度
[u,v] = pol2cart(theta,r);              %将极坐标转换为直角坐标
feather(u,v);                           %羽毛图的绘制
title('羽毛图的绘制')
```

执行上述程序，显示如图 5.51 所示的图形。

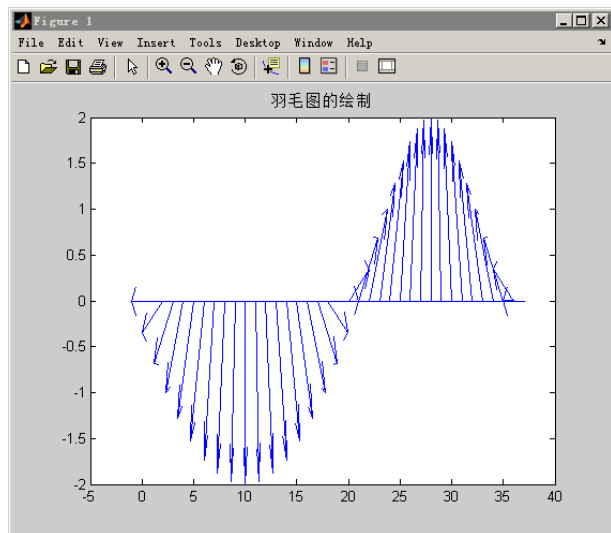


图 5.51 羽毛图的绘制

5.2.9 矢量图

函数 `quiver()` 用于在二维平面上绘制矢量图，矢量图通常和其他图形一起使用，用于显示数据的方向，其调用格式如下。

- `quiver(x,y,u,v)`: 输入参数 `x`、`y` 用于指定绘制矢量的位置，`u`、`v` 用于指定绘制的矢量在水平和垂直方向的大小。
- `quiver(u,v)`: 绘制矢量图，矢量位置采用默认设置。

【例 5.28】 矢量图的绘制。

```
[x,y] = meshgrid(-2:.2:2,-1:.15:1);
z = x .* exp(-x.^2 - y.^2);
[px,py] = gradient(z,0.2,0.15);
quiver(x,y,px,py)                    %绘制二维矢量图
title('矢量图的绘制')
```

执行上述程序，显示如图 5.52 所示的图形。

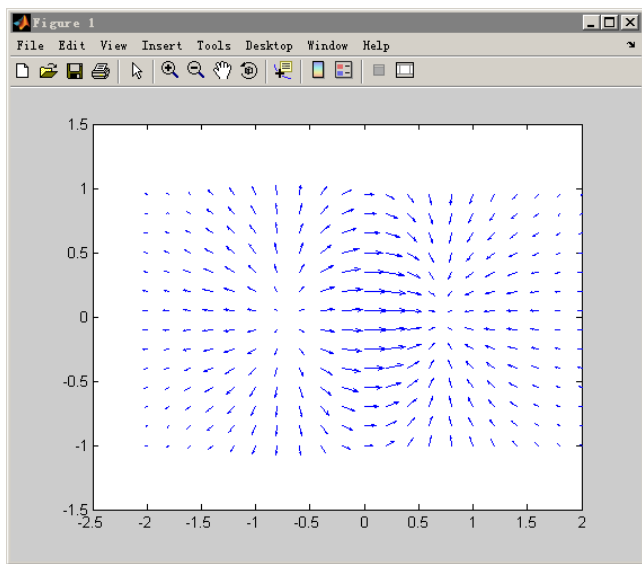


图 5.52 矢量图的绘制

5.2.10 杆型图

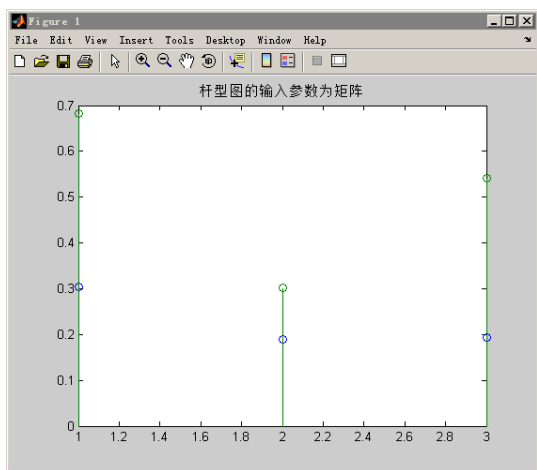
杆型图主要用来表示离散数据的变化规律，以离散的圆点表示每个数据点，并用线段把数据点和坐标轴连接起来，形如杆型。而 `plot` 函数默认则把离散的数据点间用线段连接起来。MATLAB 提供了函数 `stem()` 用于绘制杆型图，其调用格式如下。

- `stem(y)`: 绘制离散数据 `y` 的杆型图，横坐标为默认，如果数据 `y` 为向量，即最后绘制的杆型图为数据 `y` 在 `x` 轴上等间距排列的杆型，横坐标为 `1:length(y)`，如果 `y` 为矩阵，则横坐标为矩阵的行，同一行中的元素绘制在相同的横坐标下，表现为两条杆线。
- `stem(x,y)`: 指定横坐标 `x`，绘制数据 `y` 的杆型图。
- `stem(...,'fill')`: 杆型图中表示数据的点设置为填充。
- `stem(...,LineStyle)`: 设置杆型图杆型的线条。

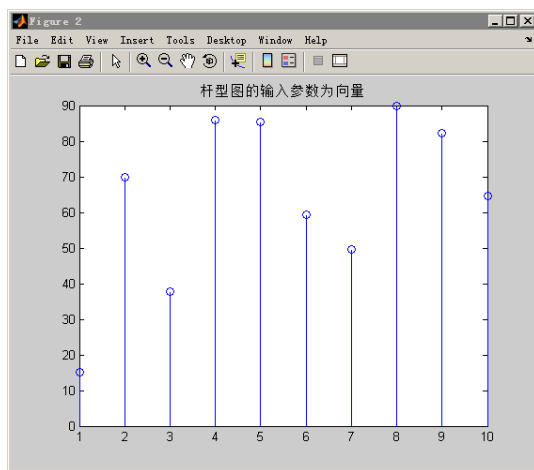
【例 5.29】 杆型图的绘制。

```
close all;
y=rand(3,2);
stem(y);           %绘制向量 y 的杆型图
title('杆型图的输入参数为矩阵')
y=rand(1,10)*100;
figure;
stem(y);           %绘制矩阵 y 的杆型图
title('杆型图的输入参数为向量')
y=rand(1,10)*100;
figure;
stem(y,'fill');    %对杆型图末端的数据点填充
title('杆型图的末端的数据点填充')
```

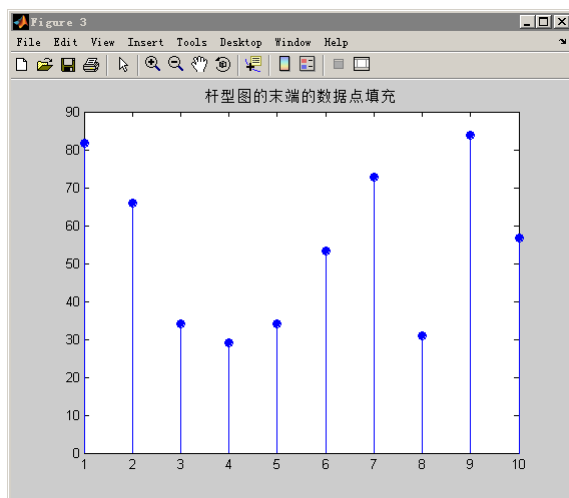
执行上述程序，显示如图 5.53 所示的图形。



(a)



(b)



(c)

图 5.53 杆型图的绘制

5.2.11 阶梯图

函数 `stairs()` 用于绘制阶梯图，可用于表示其调用格式如下。

- `stairs(y)`: 绘制数据 y 的阶梯图。如果 y 为向量，则绘制数据 y 中每个元素的阶梯变化图，横坐标 x 为 1 到 `length(y)`，如果 y 为矩阵，则对数据 y 的每一行画一阶梯图，横坐标 x 为从 1 到 y 的列数。
- `stairs(x,y)`: 指定横坐标 x 绘制阶梯图。
- `stairs(...,LineStyle)`: 参数 `LineStyle` 设置阶梯图中线条的线型、标记符号和颜色等。

【例 5.30】 阶梯图的绘制。

```
y=rand(1,10);
stairs(y);
title('输入数据为向量的阶梯图的绘制');
y=rand(3,5);
figure;
stairs(y);
title('输入数据为矩阵的阶梯图的绘制');
```

执行上述程序，显示如图 5.54 所示的图形。

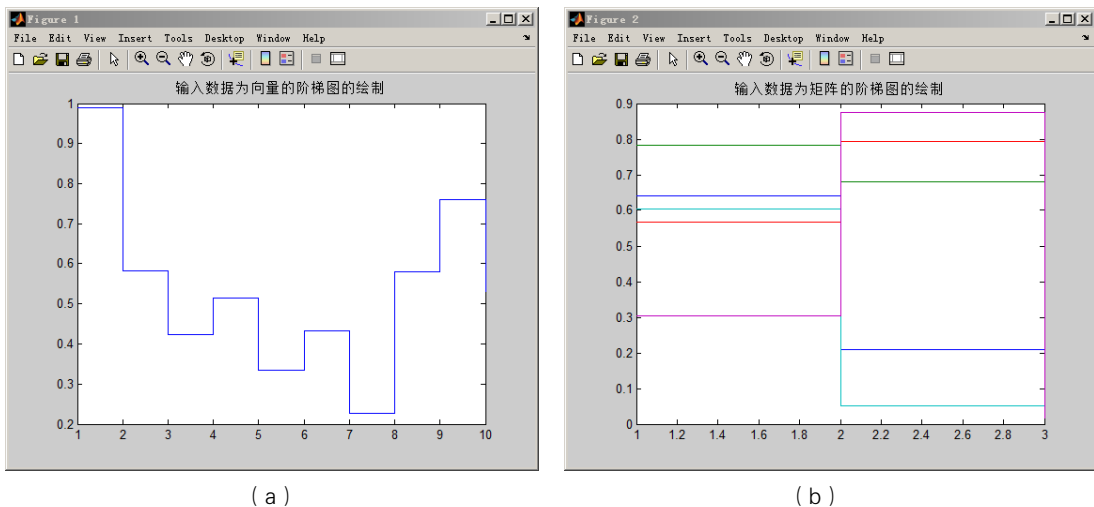


图 5.54 阶梯图的绘制

5.2.12 极坐标图

函数 `polar()` 用于绘制极坐标图，在笛卡儿坐标系平面上绘制该函数，且画出极坐标形式的栅格，其调用格式如下。

- `polar(theta,rho)`: 输入参数 `theta` 为从 `x` 轴到半径的单位为弧度的向量，`rho` 为各数据点到极点的半径向量。
- `polar(theta,rho,LineStyle)`: 参数 `LineStyle` 设置极坐标图中线条的线型、标记符号和颜色等。

【例 5.31】 极坐标图的绘制。

```
theta=0:2*pi/100:2*pi;
rho=cos(4*theta);
polar(theta, rho);
title('极坐标图的绘制');
```

执行上述程序，显示如图 5.55 所示的图形。

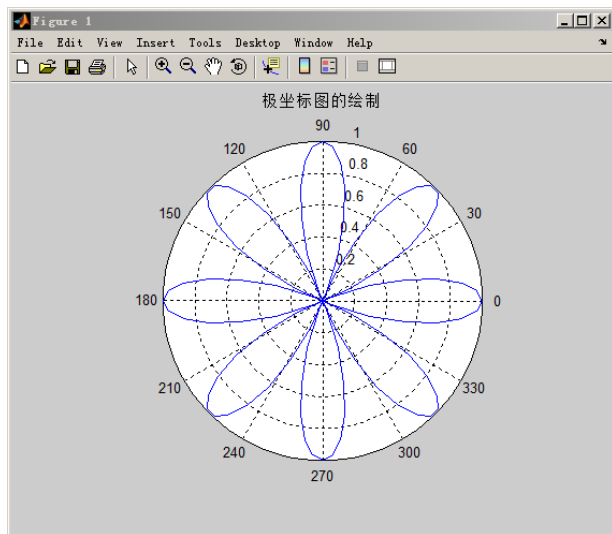


图 5.55 极坐标图的绘制

5.2.13 等值线图

等值线图可用于绘制地理数据中的等高图、气象数据中的等势图等。等值线图在二维图形中把第三维中相同大小的数据连接为等值线，一定程度上可以表示第三维的信息，同时，等值线图相比三维图更容易观察数据之间的关系，被广泛地应用于各个领域。

MATLAB 中提供了一系列的函数用于绘制不同形式的等高线图，其中包括以下函数。

1. contour()函数

contour()函数可用于绘制二维等值线图，函数的调用格式如下。

- contour(z): 输入数据 z 为二维矩阵，绘制数据 z 的等值线，绘图时等值线的数量和数值根据矩阵 z 的数据范围自动确定。
- contour(z,n): 绘制等值线图，设置等值线数目为 n。
- contour(z,v): 绘制等值线图，向量 v 设置等值线的数值。
- contour(x,y,z): 绘制矩阵 z 的等值线图，输入参数 x、y 用于指定绘制的等值线图的坐标轴数据，同时输入数据 x、y、z 必须为大小相等的矩阵。
- contour(x,y,z,n): 为指定坐标轴的等值线图设置等值线的数目 n。
- contour(x,y,z,v): 为指定坐标轴的等值线图设置等值线的数值 v。
- contour(...,LineStyle): 输入参数 LineSpec 用于设置等值线的线型。
- [c,h] = contour(...): 返回 contour()函数绘制的等高线图上的等值线的数值标签 c 和包含所有图形对象的句柄 h。

2. contourf()函数

contourf()函数用于绘制带填充的二维等值线图。即在 contour()函数绘制的等值线图上，将相邻的等值线之间用同一种颜色填充，不相邻的等值线之间填充不同的颜色，填充用的颜色决定于当前的色图颜色。函数 contourf()的调用格式同 contour()。

3. clabel()函数

- clabel(c,h): 在句柄 h 指定的等值线图上的等值线上添加数据标签 c。
- clabel(c,h,v): 在指定的等值线值 v 上显示数据标签 c。
- clabel(c,h,'manual'): 手动方式设置等值线的数据标签。当运行该命令后，等值线图中将出现十字连线，用户用鼠标左键或空格键在最接近指定位置上放置数据标签，按回车键结束该操作。
- clabel(c): 在当前的等值线图上添加数据标签 c。
- clabel(c,v): 在当前的等值线图上添加数据标签 c，并指定数据标签所加的等值线值 v。
- clabel(c,'manual'): 用户手动方式为当前等值线图添加数据标签。

另外，函数 ezcontour()和 ezcontourf()可以直接绘制函数表达式的等值线图，感兴趣的读者可以查阅 MATLAB 的相关帮助文档。

【例 5.32】等值线图的绘制。

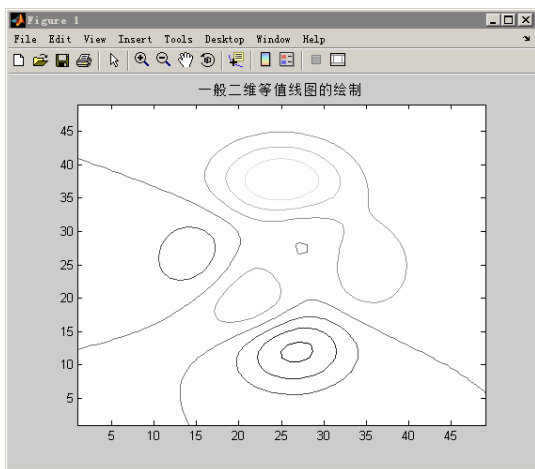
```
z=peaks; %函数 peaks 用于生成图形绘制的示例数据
[c,h] = contour(z); %生成数据矩阵 z 的不带填充的二维等值线图
title('一般二维等值线图的绘制');
clabel(c,h); %为二维等值线图添加数据标签
colorbar %添加等值线图的颜色条
title('一般二维等值线图的绘制（添加数据标签）');
figure;
v=[min(z(:)):2:max(z(:))];
[c,h] = contourf(z,v); %绘制带填充的二维等值线图，并设置等值线向量 v
```

```

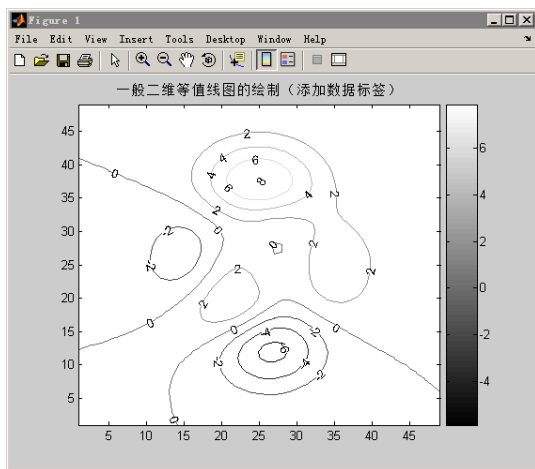
clabel(c,h); %为二维等值线图添加数据标签
colorbar %添加等值线图的颜色条
title('带填充的二维等值线图的绘制')
figure;
[c,h] = contourf(z,5); %绘制带填充的二维等值线图，并设置等值线条数为 5
title('二维等值线图手动添加等值线标签')
clabel(c,h,'manual'); %为二维等值线图手动添加数据标签

```

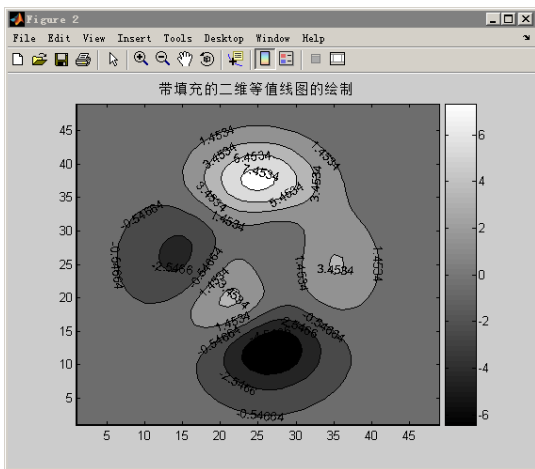
执行上述程序，显示如图 5.56 所示的图形。



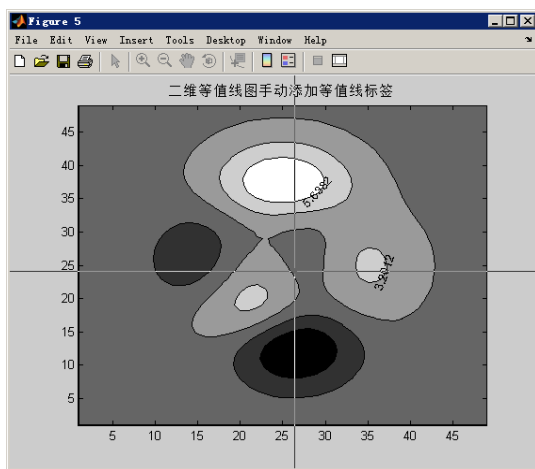
(a)



(b)



(c)



(d)

图 5.56 等值线图的绘制

5.2.14 曲线误差的添加

如果已知数据资料的误差量，就可用函数 `errorbar()` 绘制曲线，其调用格式如下。

- `errorbar(Y,E)`: 以默认的横坐标绘制数据 y 的变化图，同时在数据 y 点的上、下方各绘制一段误差线，误差线的长度为相应各点输入参数中的误差 E 。
- `errorbar(X,Y,E)`: 以指定的横坐标 x 绘制误差线。
- `errorbar(X,Y,L,U)`: L 和 U 分别指定曲线相应数据点向上的误差线和向下的误差线。
- `errorbar(...,LineStyle)`: 输入参数 `LineStyle` 用于指定线型、标记符、颜色等绘制误差线。

【例 5.33】曲线误差形图的绘制。

```
x = 0:pi/10:pi;
y = sin(x);
e = std(y)*ones(size(x));
errorbar(x,y,e)
```

执行上述程序，显示如图 5.57 所示的图形。

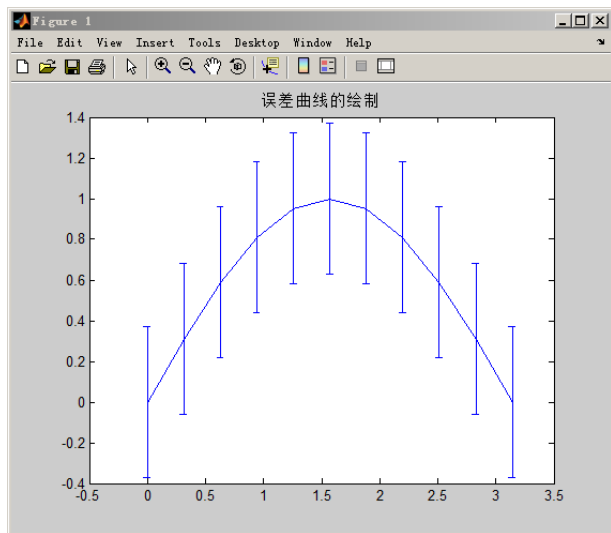


图 5.57 误差曲线的绘制

5.3 三维图形

在上一节中主要介绍了二维图形的绘制，但在有些信息量比较丰富的情况下，往往要绘制更为复杂的三维图形，三维图形信息丰富，能更好地反映数据之间的相互规律。但一般情况下三维图形的绘制比较复杂，而 MATLAB 为我们提供了函数可以直接绘制漂亮的三维图形，同时还提供了完整的三维图形编辑的功能。与一般的统计绘图软件相比，在三维图形绘制和编辑方面，MATLAB 软件更有优势。

本章将重点介绍 MATLAB 在三维图形绘制中的基础知识，包括三维图绘制和编辑的相关知识。

5.3.1 三维图形的绘制

1. 三维曲线的绘制

MATLAB 中提供了函数 `plot3()` 用于绘制三维曲线，其函数的用法与二维曲线绘制函数 `plot()` 类似。函数 `plot3()` 的调用格式如下。

- `plot3(x1,y1,z1)`: 在三维空间中绘制以 x 、 y 、 z 为坐标轴的曲线，曲线由 $x1$ 、 $y1$ 、 $z1$ 中的元素确定， $x1$ 、 $y1$ 、 $z1$ 为向量或矩阵。
- `plot3(x1,y1,z1,LineSpec,...)`: 输入参数 `LineSpec` 用于指定绘制的三维曲线的线型、标记符、颜色等。
- `plot3(x1,y1,z1, x2,y2,z2,...)`: 在三维空间内绘制多条曲线。
- `plot3(...,'PropertyName',PropertyValue,...)`: 设置由 `plot3()` 函数绘制的三维曲线的各种属性的属性值。

【例 5.34】 三维曲线图的绘制。

```

close all;
t=0:pi/10:2*pi;
x=2*t;
y=sin(t);
z=cos(t);
plot3(x,y,z);           %三维曲线图的绘制

```

执行上述程序，显示如图 5.58 所示的图形。

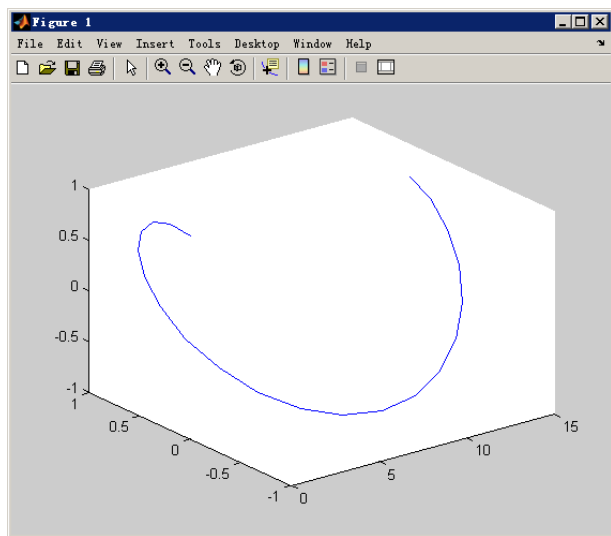


图 5.58 三维曲线图的绘制

2. 三维网格图的绘制

在三维网格图绘制时经常需要用到函数 `meshgrid()`，用于生成网格数据，其调用格式如下。

- `[X,Y] = meshgrid(x,y)`：用于生成向量 `x` 和 `y` 的网格数据，即变换为矩阵数据 `X` 和 `Y`，矩阵 `X` 中的行向量为向量 `x`，矩阵 `Y` 的列向量为向量 `y`，矩阵 `X` 和 `Y` 的大小为 `size(y) × size(x)`。
- `[X,Y] = meshgrid(x)`：生成向量 `x` 的网格数据，函数等同于 `[X,Y] = meshgrid(x,x)`。
- `[X,Y,Z] = meshgrid(x,y,z)`：生成向量 `x`、`y`、`z` 的三维网格数据。

【例 5.35】 `meshgrid()` 函数的使用。

```
[X,Y] = meshgrid(1:4,2:2:10)
```

运行上述命令，在命令窗口生成网格数据：

```

X =
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
     1     2     3     4
Y =
     2     2     2     2
     4     4     4     4
     6     6     6     6
     8     8     8     8
    10    10    10    10

```

生成的数据 `X` 和 `Y` 可分别表示三维绘图中的 `x` 和 `y` 坐标。

三维网格图形是指在三维空间内连接相邻数据点，形成网格。在 MATLAB 中绘制三维网格图的函数主要有 `mesh()`、`meshc()` 和 `meshz()`。其中，

函数 mesh() 的调用格式如下。

- mesh(x,y,z): 绘制三维网格图, x、y、z 分别表示三维网格图形在 x 轴、y 轴和 z 轴的坐标, 图形的颜色由矩阵 z 决定。
- mesh(Z): 绘制三维网格图, 分别以矩阵 Z 的列下标、行下标作为三维网格图的 x 轴、y 轴的坐标, 图形的颜色有矩阵 Z 决定。
- mesh(...,C): 输入参数 C 用于控制绘制的三维网格图的颜色。
- mesh(...,'PropertyName',PropertyValue,...): 设置三维网格图的指定属性的属性值。

函数 meshc() 可绘制带有等值线的三维网格图, 其调用格式与函数 mesh() 基本相同, 但函数 meshc() 不支持对图形网格线或等高线指定属性的设置。

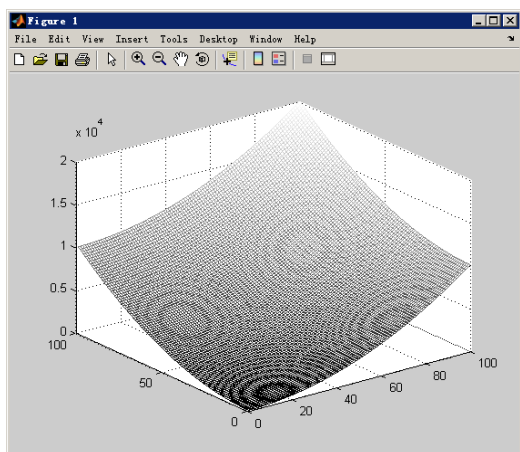
函数 meshz() 可绘制带有图形底边的三维网格图, 其调用格式与函数 mesh() 基本相同, 但函数 meshz() 不支持对图形网格线指定属性的设置。

另外, 函数 ezmesh()、ezmeshc()、ezmeshz() 可根据函数表达式直接绘制相应的三维网格图。

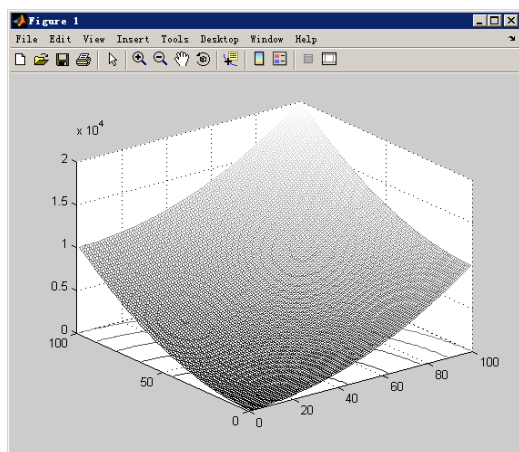
【例 5.36】三维网格图的绘制。

```
[X,Y] = meshgrid(1:1:100);
Z=X.^2+Y.^2;
mesh(X,Y,Z);           %三维网格图的绘制
meshc(X,Y,Z);          %带等高线的三维网格图的绘制
meshz(X,Y,Z);          %带图形底边的三维网格图的绘制
```

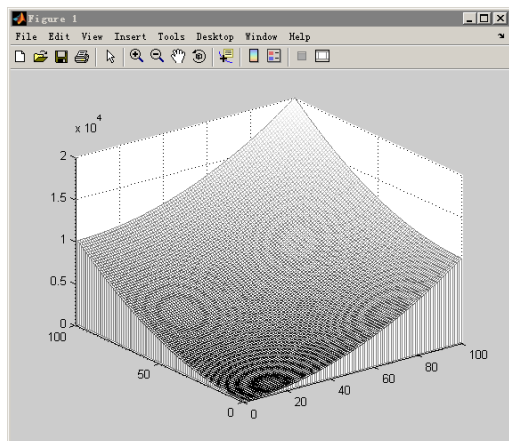
执行上述程序, 显示如图 5.59 所示的图形。



(a)



(b)



(c)

图 5.59 三维网格图的绘制

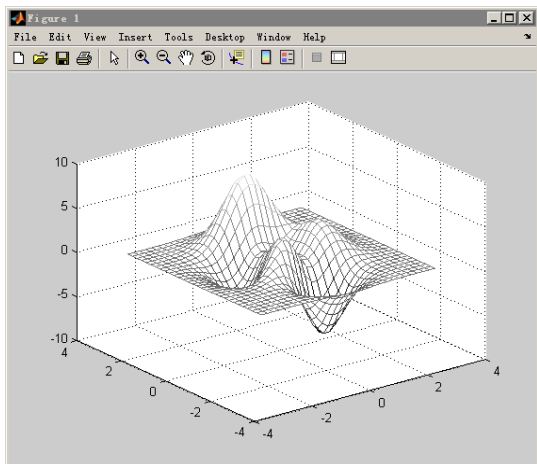
由于网格线是不透明的，绘制的三维网格图有时只能显示前面的图形部分，而后面的部分可能被网格线遮住了，没有显示出来。MATLAB 中提供了命令 `hidden` 用于观察图形后面隐藏的网格，函数 `hidden` 的调用格式如下。

- `hidden on`: 设置网格隐藏部分不可见，默认情况下为此状态。
- `hidden off`: 设置网格的隐藏部分可见。
- `hidden`: 该命令用于切换网格的隐藏部分是否可见。

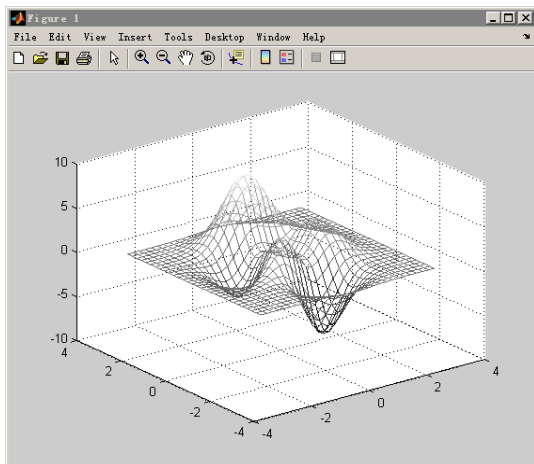
【例 5.37】 三维网格图的网格线的控制。

```
[X,Y,Z] = peaks(30);      %生成绘制三维网格图形的数据
mesh(X,Y,Z)               %绘制三维网格图
hidden off;                %控制三维网格图隐藏的网格可见
hidden;                    %切换三维网格图隐藏的网格到不可见
```

执行上述程序，显示如图 5.60 所示的图形。



(a)



(b)

图 5.60 三维网格图网格线控制

3. 三维表面图的绘制

三维表面图也可以用来表示三维空间内数据的变化规律，与之前讲述的三维网格图的不同之处在于对网格的区域填充了不同的色彩。在 MATLAB 中绘制三维表面图的函数主要有 `surf()`、`surfc()` 和 `surf1()`。其中函数 `surf()` 的调用格式如下。

- `surf(Z)`: 绘制数据 `Z` 的三维表面图，分别以矩阵 `Z` 的列下标、行下标作为三维网格图的 `x` 轴、`y` 轴的坐标，图形的颜色由矩阵 `Z` 决定。
- `surf(X,Y,Z)`: 绘制三维表面图，`X`、`Y`、`Z` 分别表示三维网格图形在 `x` 轴、`y` 轴和 `z` 轴的坐标，图形的颜色由矩阵 `Z` 决定。
- `surf(X,Y,Z,C)`: 绘制三维表面图，输入参数 `C` 用于控制绘制的三维表面图的颜色。
- `surf(...,'PropertyName',PropertyValue)`: 绘制三维表面图，设置相应属性的属性值。

函数 `surfc()` 用于绘制带等值线的三维表面图，其调用格式基本同函数 `surf()`。

函数 `surf1()` 可用于绘制带光照模式的三维表面图，与函数 `surf()` 和 `surfc()` 不同的调用格式如下。

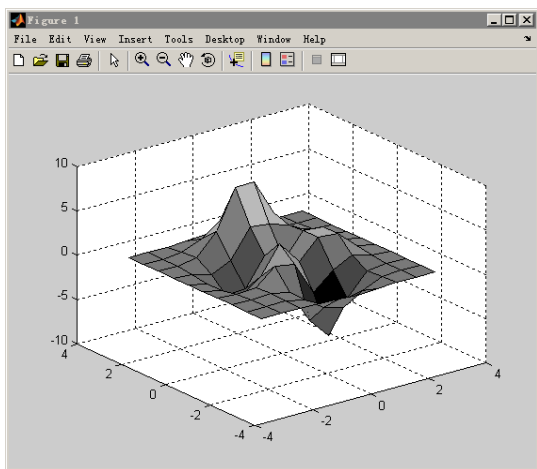
- `surf1(...,'light')`: 以光照对象 “light” 生成一个带颜色、带光照的曲面。
- `surf1(...,'cdata')`: 输入参数 `cdata` 设置曲面颜色数据，使曲面成为可反光的曲面。

- `surf1(...,s)`: 输入参数 `s` 为一个二维向量[azimuth,elevation], 或者三维向量[x, y, z], 用于指定光源方向, 默认情况下光源方位从当前视角开始, 逆时针 45 度。

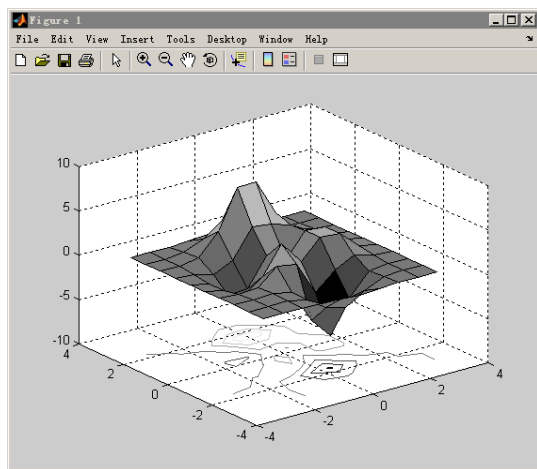
【例 5.38】 三维表面图的绘制。

```
[X,Y,Z] = peaks(10);           %生成绘制三维表面图形的数据
surf(X,Y,Z)                   %绘制三维表面图
figure;
surfc(X,Y,Z)                  %绘制带等值线的三维表面图
figure;
surf1(X,Y,Z)                  %绘制考虑光照效果的三维表面图
```

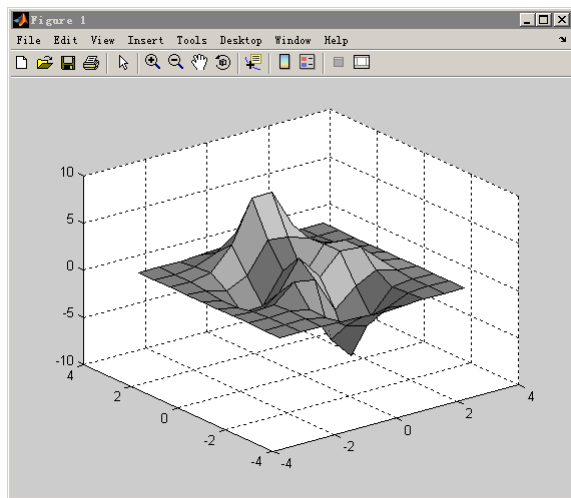
执行上述程序, 显示如图 5.61 所示的图形。



(a)



(b)



(c)

图 5.61 三维表面图的绘制

4. 三维等值线的绘制

函数 `contour3()` 可用于绘制三维等值线图, 即在三维栅格图形上进一步添加等值线。类似于二维等值线图绘制函数 `contour()`, 其调用格式如下。

- `contour3(z)`: 绘制矩阵 `z` 的三维等值线图, 横纵坐标默认为矩阵的列下标和行下标。
- `contour3(z,n)`: 绘制矩阵 `z` 的三维等值线图, `n` 用于指定等值线的数目。

- `contour3(z,v)`: 矩阵 `z` 的三维等值线图, 向量 `v` 用于指定各等值线的值。
- `contour3(x,y,z)`: 绘制矩阵 `z` 的三维等值线图, 输入参数 `x`、`y` 用于指定绘制的等值线图的坐标轴数据, 同时输入数据 `x`、`y`、`z` 必须为大小相等的矩阵。
- `contour3(x,y,z,n)`: 为指定坐标轴的三维等值线图设置等值线的数目 `n`。
- `contour3(x,y,z,v)`: 为指定坐标轴的三维等值线图设置等值线的数值 `v`。
- `contour3(...,LineStyle)`: 输入参数 `LineStyle` 指定的等值线的线型与颜色。
- `[c,h] = contour3(...)`: 返回 `contour3()` 函数绘制的三维等值线图上的等值线的数值标签 `c` 和图形对象句柄 `h`。

【例 5.39】三维等值线图的绘制。

```
[X,Y] = meshgrid([-2:.25:2]);
Z = X.*exp(-X.^2-Y.^2);
contour3(X,Y,Z,10)           %绘制三维等值线图, 并指定等值线的条数为 10
```

执行上述程序, 生成如图 5.62 所示的图形。

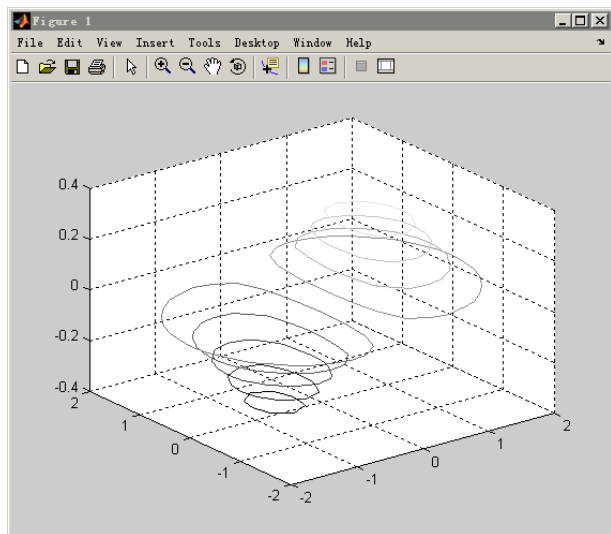


图 5.62 三维表面图的绘制

5. 三维切片图的绘制

函数 `slice()` 用于绘制三维切片图。三维切片图可形象地称为“四维图”, 可以在三维空间内表达第四维的信息, 用颜色来标识四维数据的大小。`slice()` 函数的调用格式如下。

- `slice(v,sx,sy,sz)`: 输入参数 `v` 为三维矩阵 (阶数为 $m \times n \times p$), `x`、`y`、`z` 轴默认状态下分别为 `1:m`、`1:n`、`1:p`, 数据 `v` 用于指定第四维的大小, 在切片图上显示为颜色的不同, 输入参数 `sx`、`sy`、`sz` 分别用于指定切片图在 `x`、`y`、`z` 轴所切的位置。
- `slice(x,y,z,v,sx,sy,sz)`: 输入参数 `x`、`y`、`z` 用于指定绘制的三维切片图的 `x`、`y`、`z` 轴。
- `slice(...,'method')`: 输入参数 `method` 用于指定切片图绘制时的内插值法, 'method' 可以设置的参数有: 'linear' (三次线性内插值法, 默认)、'cubic' (三次立方内插值法)、'nearest' (最近点内插值法)。

【例 5.40】三维切片图的绘制。

```
[x,y,z] = meshgrid(-2:.25:2,-2:.25:2,-2:.16:2);
v = x.*exp(-x.^2-y.^2-z.^2);
xslice = [-1.2,.8,2];
yslice = 2;
zslice = [-2,0];
```

%指定三维切片图的 x、y、z 轴数据
 %第四维数据 v
 %三维切片图在 x 轴方向上的切片位置
 %三维切片图在 y 轴方向上的切片位置
 %三维切片图在 z 轴方向上的切片位置

```
slice(x,y,z,v,xslice,yslice,zslice)
```

%绘制三维切片图

执行上述程序，生成如图 5.63 所示的三维切片图。

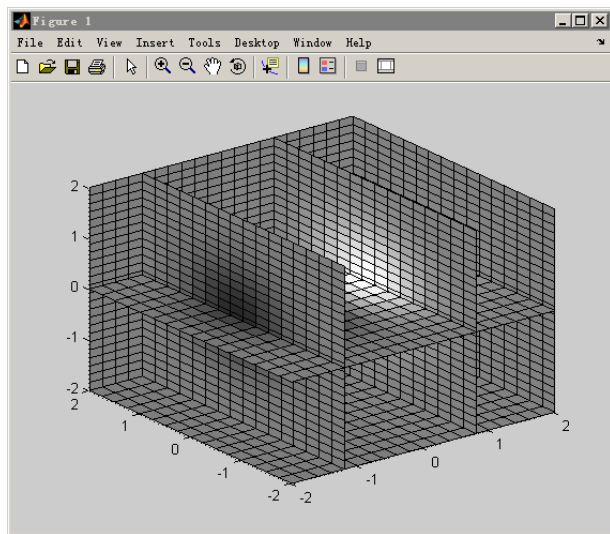


图 5.63 三维切片图的绘制

6. 其他一些三维图形的绘制

由于一些特殊的需要，用户可能需要绘制一些具有一定形状的三维图，例如瀑布图、柱面图、球体等，下面简单演示这些图形的绘制。

(1) waterfall(): 瀑布图。

函数 waterfall() 可以绘制在 x 轴或者 y 轴方向具有流水效果的瀑布图，其调用格式如下。

waterfall(x,y,z): 输入参数 x、y 与 z 分别用于指定瀑布图的 x 轴，y 轴和 z 轴，同时数据 z 标识瀑布图的颜色。

【例 5.41】瀑布图的绘制。

```
[x,y,z]=peaks(25); %瀑布图绘制的数据生成
waterfall(x,y,z); %瀑布图绘制
```

执行上述程序，显示如图 5.64 所示的瀑布图。

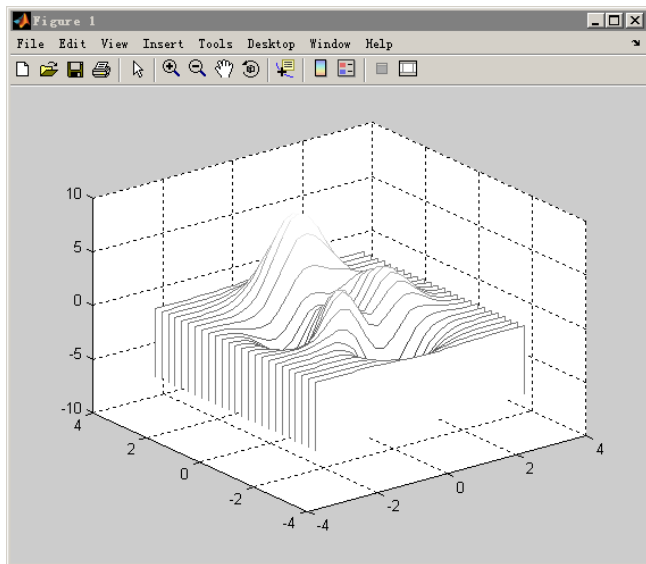


图 5.64 瀑布图的绘制

(2) `cylinder()`: 柱面图。

函数 `cylinder()` 可生成关于 z 轴旋转对称的柱面体, 结合函数 `surf` 或 `mesh` 可生成柱面体的三维曲面图, 其调用格式如下。

- `[x,y,z] = cylinder`: 返回半径为 r , 高度为 1 的圆柱体的 x 、 y 、 z 轴的坐标值, 圆柱一周的分点数为 20。
- `[x,y,z] = cylinder(r)`: 返回半径为 r , 高度为 1 的圆柱体的 x 、 y 、 z 轴的坐标值, 圆柱一周的分点数为 20。
- `[x,y,z] = cylinder(r,n)`: 返回半径为 r , 高度为 1 的圆柱体的 x 、 y 、 z 轴的坐标值, 圆柱一周的分点数为 n 。

【例 5.42】 柱面图的绘制。

```
t=0:pi/10:pi;
r=cos(t);
[x,y,z] = cylinder(r);           %生成半径为 5 的圆柱体的坐标
mesh(x,y,z*10);                 %绘制圆柱体的三维曲面图
```

执行上述程序, 显示如图 5.65 所示的柱面图。

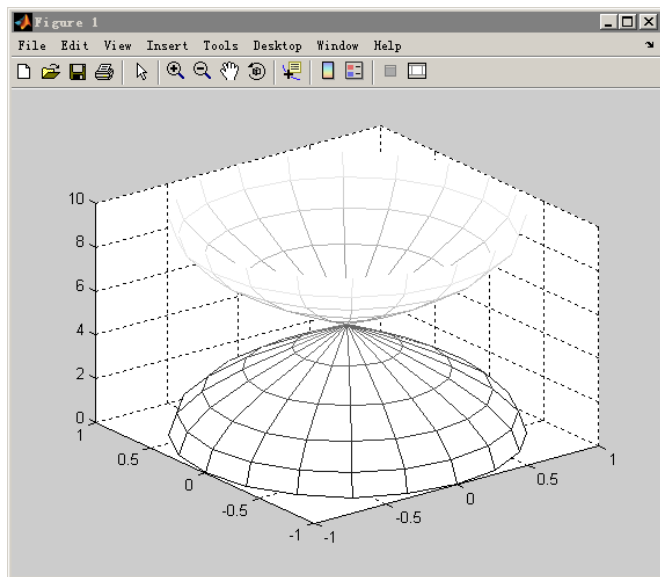


图 5.65 柱面图的绘制

(3) `sphere()`: 球形图。

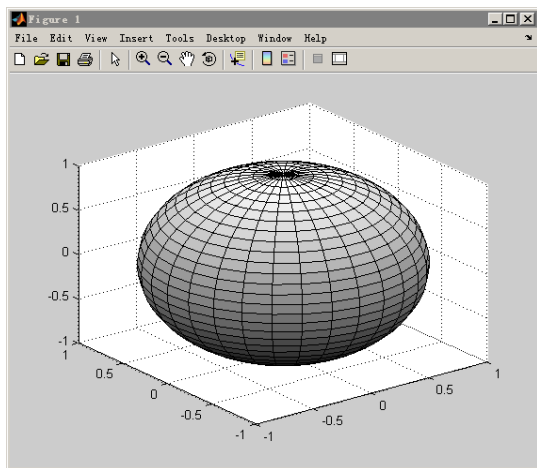
函数 `sphere()` 用于在直角坐标系内绘制球形图, 其调用格式如下。

- `sphere(n)`: 绘制单位球形图, 球体包含 $n \times n$ 个球面。
- `[x,y,z] = sphere(n)`: 返回含 $n \times n$ 个球面的球体的坐标, 结合函数 `surf` 或 `mesh` 绘制三维球体曲面图。

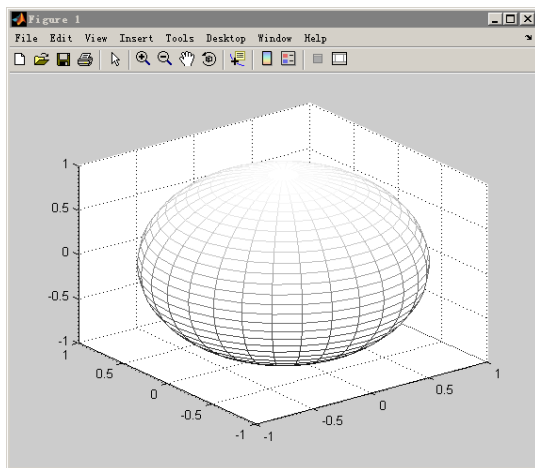
【例 5.43】 球形图的绘制。

```
sphere(30);                     %直接绘制包含 30×30 个球面的球形图
[x,y,z]=sphere(30);             %返回包含 30×30 个球面的球形图的坐标
figure;
mesh(x,y,z);                     %绘制三维球形网格图
```

执行上述程序, 显示如图 5.66 所示的球形图。



(a)



(b)

图 5.66 球形图的绘制

(4) ellipsoid(): 椭球形图。

函数 ellipsoid() 可生成绘制椭球体图的坐标数据, 结合函数 surf 或者 mesh, 即可绘制三维椭球体图, 其函数的调用格式如下。

- $[x,y,z] = \text{ellipsoid}(xc, yc, zc, xr, yr, zr, n)$: 输入参数中 xc 、 yc 、 zc 为椭球体的球心坐标, xr 、 yr 、 zr 为椭球体 3 个半轴的长度, n 为椭球体的分点数。
- $[x,y,z] = \text{ellipsoid}(xc, yc, zc, xr, yr, zr)$: 输入参数 xc 、 yc 、 zc 、 xr 、 yr 、 zr (作用同上), 椭球体的分点数默认为 20。

【例 5.44】椭球形图的绘制。

```
[x,y,z] = ellipsoid(1,1,1,2,3,4); %返回椭球体的坐标
surf(x,y,z); %绘制三维椭球体的曲面图
```

执行上述程序, 显示如图 5.67 所示的椭球形图。

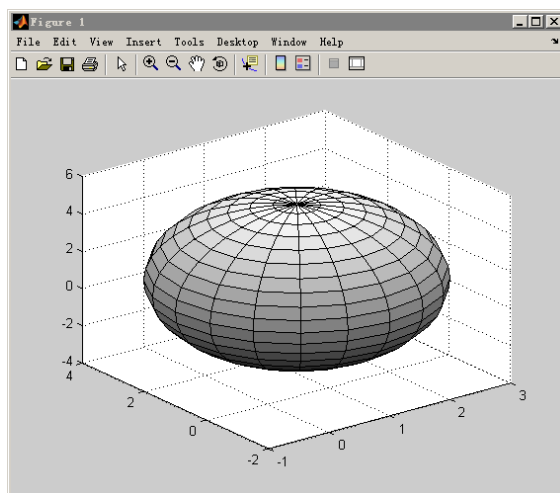




图 5.67 椭球形图的绘制

5.3.2 三维图形的编辑

三维图形比二维图形信息更为丰富, 但是图形本身也更为复杂, 为了更好地获得三维图形上

的信息，我们往往需要设置三维图形的属性，例如观察的视角、图形的颜色等，以期通过三维图像获取更多的信息。本小节将介绍三维图形如何编辑处理，包括图形的视角控制、颜色控制、光线控制，以指导用户进一步完善绘制的三维图形。

1. 图形的视角控制

对于三维图形采用不同的角度观察都会获得不同的信息，MATLAB 中通过对图形视角的控制，可以让一幅图形显示出从不同角度观察的效果。视角的控制可以通过函数 `view()` 设置或者直接在图形窗口中利用工具栏图标调整观察图形的角度，选中图标，在图形上方通过鼠标控制，可方便地旋转当前窗口的三维图形，但是函数 `view()` 能够更为精确地设置观察图形的角度，其调用格式如下。

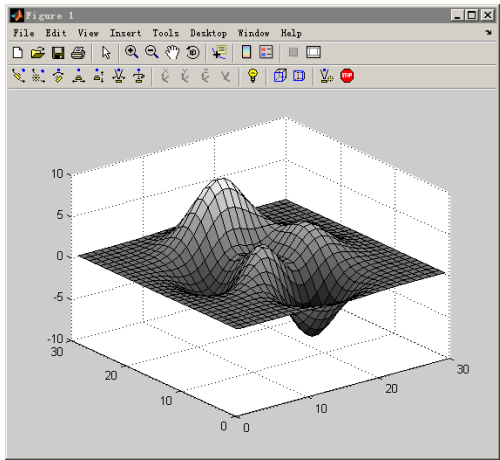
- `view(az,el) / view([az,el])`: 设置观察图形的视角，其中输入参数 `az` 为观察图形的方位角，方位角为 x - y 轴平面内的平面角，逆时针方向角度为正，`el` 为观察图形的俯仰角，俯仰角为视点相对 x - y 轴平面的角度，沿 z 轴正方向仰起的角度为正；参数 `az` 和 `el` 的单位为度，默认情况下三维图形的观察视角为 `az=-37.5`，`el=0`。
- `view([x,y,z])`: 输入参数为观察点在三维图形坐标体系内的坐标 x 、 y 、 z ，换算视角为 `az=atan(-x/y)*(180/pi)`，`el=atan(z/sqrt(x^2+y^2))*(180/pi)`。
- `view(2)`: 以二维图形的观察视角 `az=0`，`el=90` 观察图形。
- `view(3)`: 以三维图形的默认观察视角 `az=-37.5`，`el=0` 观察图形。
- `[az,el] = view`: 返回当前窗口图形的视角。

【例 5.45】图形视角的绘制

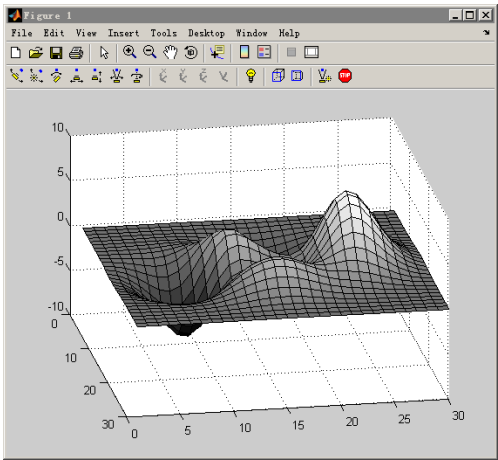
```
surf(peaks(30));           %绘制三维曲面图
[az,el]=view              %返回当前视场的角度
view(80,30);              %设置图形的观察视角，方位角 80 度，俯仰角 30 度
```

执行上述程序，命令窗口显示如下所示的视场角度数据，同时生成图 5.68 所示的不同视场下的图形。

```
az =
-37.5000
el =
30
```



(a)



(b)

图 5.68 图形视角的绘制

2. 图形的颜色控制

三维图形丰富的颜色使其具有更好的表现力，MATLAB 中的图形颜色是通过颜色映像机制来控制

制的。颜色映像机制用于产生颜色与数值之间对应的颜色表，表为三列 0~1 的数据矩阵，各列数据代表颜色中 R、G、B 分量的大小，每行数据颜色 R、G、B 分量大小不同，将产生不同的颜色。

MATLAB 已为用户提供了多种典型的颜色映像，如无特殊的需要，一般内设的颜色映像即可满足用户的需要。MATLAB 中内设的颜色映像如下。

- cool: 青蓝和洋红的颜色映像。
- bone: 带一点蓝色调的灰度颜色映像。
- flag: 交替的红色、白色、蓝色和黑色颜色映像。
- jet: hsv 的一种变形，以蓝色开始和结束，为默认状态下的颜色映像。
- copper: 线性铜色调颜色映像。
- hsv: 色彩饱和度，以红色开始，依次经过黄、绿、青、蓝、紫，最后以红色结束的颜色映像。
- hot: 从黑色到红色，再到黄色到白色的颜色映像。
- gray: 线性灰度的颜色映像。
- pink: 粉红色的颜色映像。
- prism: 三棱镜模式，交替的红色、橘黄色、黄色、绿色和天蓝色颜色映像。
- lines: 线性颜色映像。
- white: 白色的颜色映像。
- colorcube: 增强立方颜色映像。
- autumn: 秋天风格颜色，红色到黄色颜色映像。
- spring: 春天风格颜色，洋红到黄色颜色映像。
- summer: 夏天风格颜色，绿色到黄色颜色映像。
- winter: 冬天风格颜色，蓝色到绿色颜色映像。

函数 colormap() 用于图形不同颜色映像模式的控制，其调用格式如下。

- colormap(map): 设置当前图形的颜色映射模式为 map，map 可为 MATLAB 提供的典型颜色映射模式中的任意一种，或者用户根据实际需要定义的数据矩阵。
- colormap('default'): 设置当前图形的颜色映射模式为默认的“jet”。
- cmap = colormap: 参数 cmap 返回当前图形的颜色映射模式。

【例 5.46】图形颜色的控制。

```
mesh(peaks(30));           %绘制三维网格图
colormap('gray');          %颜色映射模式设置为 gray
```

执行上述程序，显示如图 5.69 所示的图形。

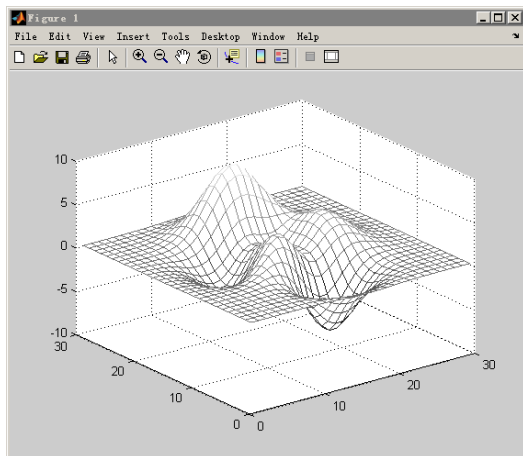


图 5.69 图形颜色的控制

在 MATLAB 中除了可以对图形颜色设置外,也可以对图形的背景色进行设置。用于图形背景色设置的函数为 `colordef()`,函数的作用对象为指定背景色后的所有图形,所以当用户需要绘制背景色为黑色的图形,需要先执行 `colordef()`函数。其调用格式如下。

- `colordef white`: 设置图形背景色为白色,默认情况下即为白色。
- `colordef black`: 设置图形背景色为黑色。
- `colordef none`: 无背景色,即此时图形的背景颜色与图形窗口的颜色相同。
- `colordef(fig,color_option)`: 设置句柄 `fig` 对应的图形的背景色为 `color_option`。

【例 5.47】 图形背景色的设置。

```
colordef black           %图形背景色设置为黑色
peaks(30);
```

执行上述程序,显示如图 5.70 所示图形。

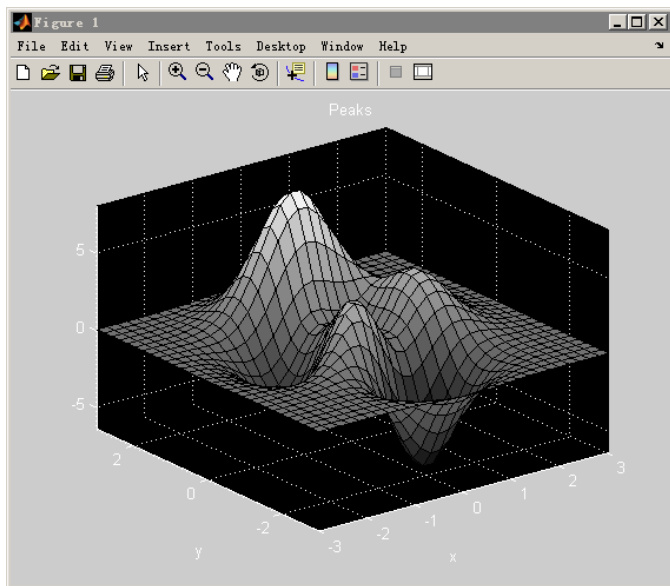



图 5.70 图形背景色的设置

函数 `colorbar()`可为彩色图形添加颜色标注,即图形上不同颜色对应的数据,同时,单击工具栏中的图标,也具有添加颜色标注的作用。函数 `colorbar()`的调用格式如下。

- `colorbar`: 添加图形标注,默认位置为图形坐标轴外部右侧。
- `colorbar('vert')`: 添加垂直的颜色标注到当前的坐标轴。
- `colorbar('horiz')`: 添加水平的颜色标注到当前的坐标轴。
- `colorbar('location')`: 添加图形标注,并设置图形标注的位置为 `location`,参数 `location` 可以设置的值: North (坐标轴内部上侧)、South (坐标轴内部下侧)、East (坐标轴内部右侧)、West (坐标轴内部左侧)、NorthOutside (坐标轴外部上侧)、SouthOutside (坐标轴外部下侧)、EastOutside (坐标轴外部右侧)和 WestOutside (坐标轴外部左侧)。

函数 `caxis()`可用于设置颜色标注的刻度范围和比例,其调用格式如下。

- `caxis([cmin cmax])`: 设置颜色标注的范围为 `cmin` 到 `cmax`,数据中小于 `cmin` 或大于 `cmax` 的,分别映射到最小值与最大值对应的颜色,而 `cmin` 与 `cmax` 之间的数据将线性地映射到当前颜色映射模式中。
- `caxis auto`: 使用系统默认的颜色标注方式,系统自动地计算数据的最大值与最小值对应的

颜色。

- `caxis manual`: 固定当前颜色标注的刻度范围, 当图像保持开启后, 可使后面的图形命令使用相同的颜色标注。

【例 5.48】颜色标注的控制。

```
surf(peaks(10));           %绘制三维曲面图
caxis([-1 1])              %设置颜色标注的刻度范围
colorbar;                  %添加颜色标注
```

执行上述程序, 显示如图 5.71 所示图形。

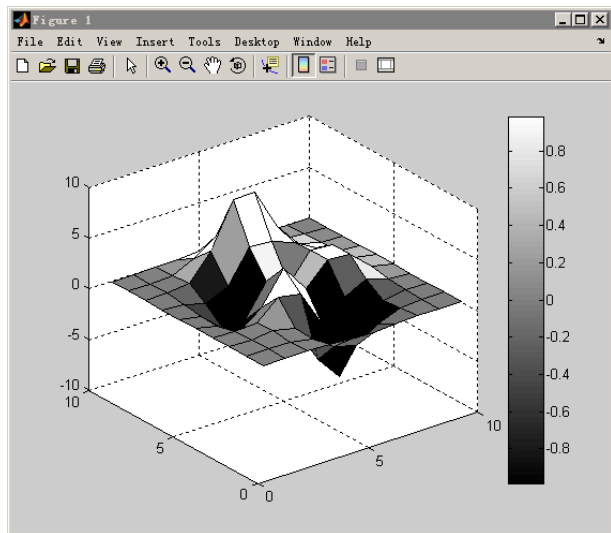


图 5.71 颜色标注的绘制

3. 图形的光线控制

通过对图形的光线控制可以使图形的逼真效果更好, MATLAB 中提供了函数 `camlight()`、`light()`、`lighting()`、`material()` 控制图形的光线效果, 下面详细介绍这些函数的使用。

(1) 函数 `camlight()`。

- `camlight headlight`: 创建灯光, 位于摄像方位的上方。
- `camlight right`: 创建灯光, 位于摄像方位的右侧。
- `camlight left`: 创建灯光, 位于摄像方位的左侧。
- `camlight`: 创建灯光, 默认情况下位于摄像方位的右侧。
- `camlight(az,el)`: 创建灯光, 输入参数 `az`、`el` 分别表示灯光的方位角和俯仰角。
- `camlight(...'style')`: 参数 `style` 用与设置创建的灯光的类型, 类型 `local` (默认) 表示创建的灯光在各个方向都有辐射光, 类型 `infinite` 表示创建的灯光其光源处于无限远处, 发射平行光。

(2) 函数 `light()`。函数 `light()` 用于创建光源对象, 其调用格式如下。

- `light('PropertyName',PropertyValue,...)`: 设置创建的光源的 `PropertyName` 属性值为 `PropertyValue`, 可以设置的属性有: `position` 属性, 表示灯光的位置; `color` 属性表示灯光的颜色; `style` 属性, 表示光源的类型。
- `handle = light(...)`: 返回创建光源对象的句柄。

(3) 函数 `lighting()`。函数 `lighting()` 用于设置光源照明的模式, 其调用格式如下。

- `lighting flat`: 光源均匀地照射目标对象表明 (默认状态)。
- `lighting gouraud`: 光源对小表面的交叉颜色插值。

- lighting phong: 光源对小表面的交叉颜色插值, 并计算像素的反射比。
 - lighting none: 关闭当前光源。
- (4) 函数 material()。函数 material() 用于控制光照效果的材质, 其调用格式如下。
- material shiny: 该材质下镜面反射较大, 图形对象比较明亮。
 - material dull: 该材质下漫散射较强, 没有镜面反射, 图形对象比较暗淡。
 - material metal: 该材质下的图形对象具有金属光泽。
 - material([ka kd ks]): 输入参数 ka、kd、ks 分别用于设置背景光、漫散射、镜面反射的强度。
 - material([ka kd ks n]): 设置背景光、漫散射、镜面反射的强度及镜面反射指数 n。
 - material([ka kd ks n sc]): 设置背景光、漫散射、镜面反射的强度、镜面反射指数 n 及镜面反射系数 sc。
 - material default: 设置光照效果的材质为默认状态。

【例 5.49】 图形的光线绘制。

```
surf(peaks(20));           %绘制三维曲面图
camlight;                  %创建光源
light('color','w','style','infinite') %设置创建光源的颜色和形式
lighting gouraud;          %设置光源的照明模式
material metal;            %设置图形的材质效果
```

执行上述程序, 显示如图 5.72~5.76 所示的各图形。

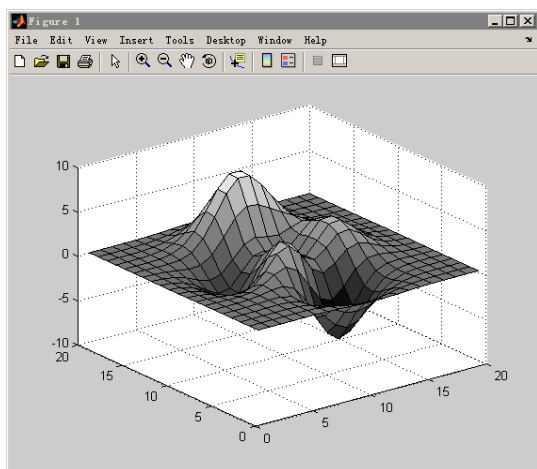


图 5.72 图形的光线绘制 (原始无光源)

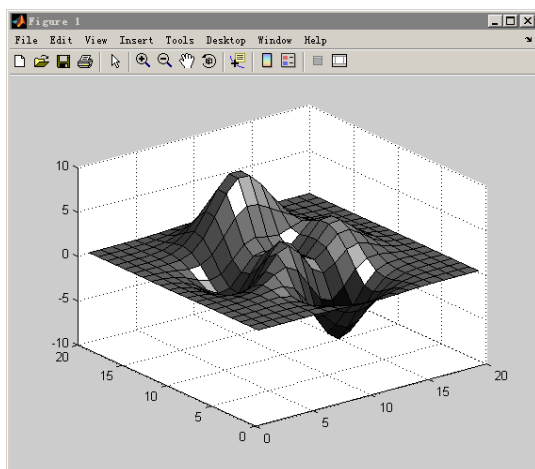


图 5.73 图形的光线绘制 (右侧创建新光源)

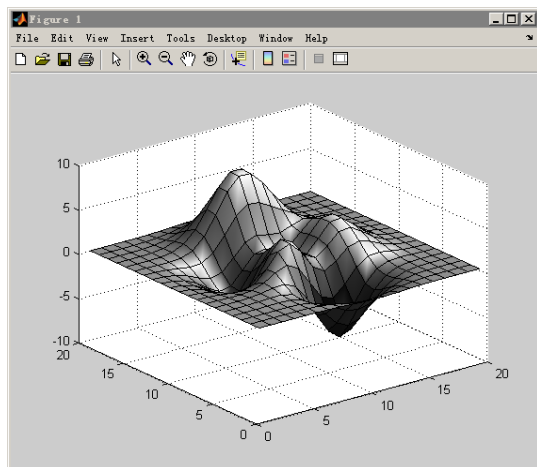
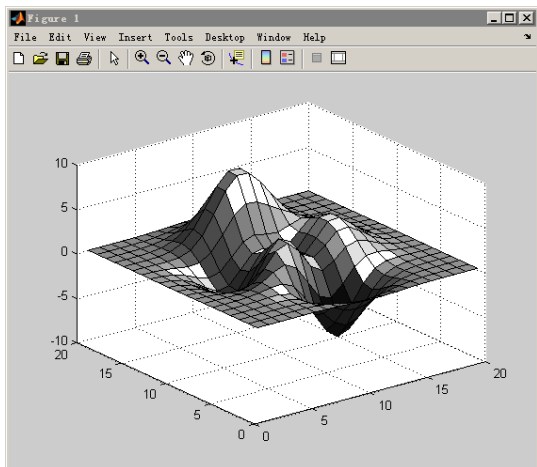


图 5.74 图形的光线绘制（设置光源颜色、类型属性后） 图 5.75 图形的光线绘制（设置光源的照明模式后）

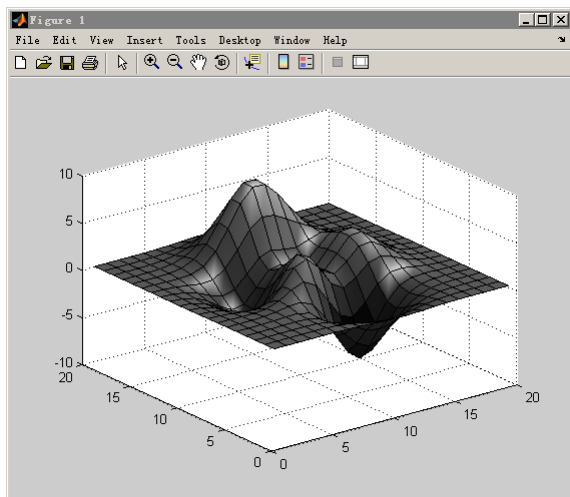


图 5.76 图形的光线绘制（设置图形的材质效果后）

5.4 本章小结

本章主要介绍了 MATLAB 图形处理的相关知识。通过本章的学习，读者可以掌握的二维、三维绘图函数，包括常用曲线图、曲面图、网格图及一些特殊图形的绘制。

利用 MATLAB 提供的绘图函数，读者可以很容易绘制出指定类型的图形，但是对于一般用户来说，往往在默认设置下绘制的图形并不能满足实际的需要，此时可以进一步地编辑完善图形。MATLAB 为我们提供了命令函数法和图形界面两种方式编辑完善图形，读者可以根据自己的喜好，灵活地编辑图形。



第6章

图形用户界面 (GUI)

图形用户界面 (GUI) 为用户和程序之间提供了良好的交互方式, 用户仅通过鼠标、键盘等简单的输入设备即可与计算机复杂的程序文件进行交互处理。利用图形用户界面设计的程序易被用户操作使用, 用户无需了解程序算法的源代码, 无须知道函数文件各输入/输出参数的意义, 直接在图形界面中按照界面设计者的要求输入相应参数, 用鼠标单击程序运行的相应按钮, 即可轻松完成复杂程序代码的执行。通过图形界面操作, 可以有效地减少用户不正确的使用程序文件所引起的程序出错。

MATLAB 提供了良好的图形用户界面设计平台, 可以设置基本的图形窗口的对象 (菜单、按钮、文本框、单选框、复选框等); 可以设置对用户不同操作的响应; 可以方便地对界面布局管理。通过 MATLAB 的图形界面设计可以方便地实现读者与程序之间的交互功能。MATLAB 中图形用户界面设计的方式主要有通过 GUI 向导创建和编写程序设计两种方式, 本章将详细介绍这两种方式如何设计图形用户界面。其中, 利用 GUI 向导创建图形界面的图形界面对于入门用户较为方便, 而图形界面控件对象功能的实现主要利用编程的方式实现, 本章主要从这两个方面向读者介绍如何在 MATLAB 中利用图形用户界面功能, 完成简单的 GUI 设计工作。

6.1 GUI 简介

图形用户界面 (Graphical User Interfaces, GUI) 是由窗口、菜单、各种控件对象构成的一个用户界面。用户通过一定的操作, 例如单击图形界面上的某个按钮, 激活 MATLAB 的图形对象, 使 MATLAB 执行相应的程序命令。

MATLAB 7.0 中的基本图形用户界面对象主要分为: uicontrol (控件对象)、uimenu (下拉式菜单对象)、uicontextmenu (弹出式菜单对象) 3 类。其中, uicontrol 对象能够创建按钮、单选框、文本框、列表框等图形用户界面对象, uimenu 对象能创建下拉式菜单和子菜单等图形用户界面对象, uicontextmenu 对象能创建弹出式菜单。对上述图形用户界面对象进行相应的设计, 即可完成交互能力强、使用方便的图形用户界面。

6.1.1 GUI 的创建方法概述

MATLAB 提供了两种方式创建 GUI 界面。如果 GUI 界面中的各种对象不是很多, 需要设置的属性也不是很复杂, 可以通过 GUIDE 向导, 利用鼠标简单拖曳完成 GUI 界面的设计工作, 相应的 GUI 界面控制执行的程序命令, 即通过回调函数使程序命令与 GUI 界面中的操作相关联; 而对于比较庞大的项目, 需要对各控件的属性精确设置, 界面中的各种控件需要频繁地编辑、修改, 此时使用编程实现相对更为方便。用户可以结合自己的需要, 使用不同的创建方法。

6.1.2 GUI 的设计流程

不管使用何种方式设计 GUI 界面，一般都要遵从以下的流程：

(1) 首先，应该明确界面设计的目标，用户需要通过该图形界面完成什么任务？需要多少界面的输入参数？计算结果通过何种方式交互的返回给用户？

(2) 按照界面设计的总体要求，合理构思界面的布局，可在演算纸上先绘制界面的草图，明确各界面之间如何衔接。

(3) 在 MATLAB 中设计图形界面，设计相应对象的参数，并力图使设计的图形界面简洁、美观、易操作。

(4) 为图形界面上控制程序执行的按钮，编写相应的回调函数。

(5) 调试 GUI 界面的执行。

6.1.3 GUI 界面设计的原则

具有相同功能的图形用户界面可能由于设计者的不同，图形界面上相差甚大。好的图形界面不但有利于用户操作使用，而且对设计者本人来说，养成良好的界面设计原则，可以减轻代码调试的开销。下面介绍主要的界面设计原则。

1. 简洁性

程序界面绝对不是越复杂越显得高级，好的程序界面是以最少的界面语言向用户表达清楚界面所能实现的任务，应该便于用户理解。尽量减少需要在不同窗口来回切换的图形界面的设计。同时，对于一些图形界面必须设计比较复杂、繁多的输入参数，可以考虑不同用户的需要，一般用户可能对一些输入参数的要求不是很高，也不是很了解一些参数的具体意义，界面中可以隐去这些参数的设置而使用默认的参数，而对于高级用户，可以通过选择一定的选项展开或显示界面中的复杂参数设置。

2. 规范性

首先在设计程序前我们必须清楚认识到 GUI 未来的用户，往往已习惯 Windows 的图形界面，为了避免不必要的麻烦，也为了用户能更快、更好地掌握基于 MATLAB 的 GUI 界面，建议 GUI 设计者参照 Windows 系统中窗口的风格、色彩的搭配、字体的设置等。

3. 合理性

优秀的 GUI 界面应该具有合理的布局，符合用户使用 GUI 界面的习惯，例如设计者应该将控制界面响应的控件放在图形界面中参数设置对话框后面。同时，设计者应该尽量保持界面的整齐、美观性。

4. 人性化

设计者在界面设计中要多为未来使用此 GUI 界面的用户着想，尽量在界面中避免生僻的专业词汇，而使用简洁明了易于被用户理解的词汇。同时在一些必要的地方可以添加醒目的操作注意事项，或者在程序执行出错后给出错误提示对话框，减少或者纠正由于用户不正确操作而引起的 GUI 界面出错的问题。另外，对于运行比较费时间的程序，可以设计显示程序运行进度的进度条，也可以为用户提供程序运行完毕自动保存结果、关机的选项等。

6.2 利用 GUIDE 设计 GUI

本节主要介绍如何利用图形用户界面向导 (Graphical User Interfaces Development Enviroment,

GUIDE) 创建 GUI 文件。GUIDE 为 GUI 的设计提供了操作简单、直观的 GUI 设计窗口，在其中可以设计常用的界面对象控件、菜单等。

6.2.1 新建 GUI 设计界面

在 MATLAB 7.0 的主界面选择“File”→“New”→“GUI”命令，弹出如图 6.1 所示的“GUIDE Quick Start”对话框，用于新建 GUI 设计界面或者打开已有的 GUI 文件，或者在命令窗口输入“GUIDE”命令也可以打开“GUIDE Quick Start”对话框。其中，“Create New GUI”选项卡可用于选择创建的 GUI 文件的模板，包括 Blank GUI (默认模板)、GUI with Uicontrols (带控件对象的 GUI 模板)、GUI with Axes and Menu (带坐标轴与菜单的 GUI 模板) 与 Modal Question Dialog (带模式问答对话框的 GUI 模板)。

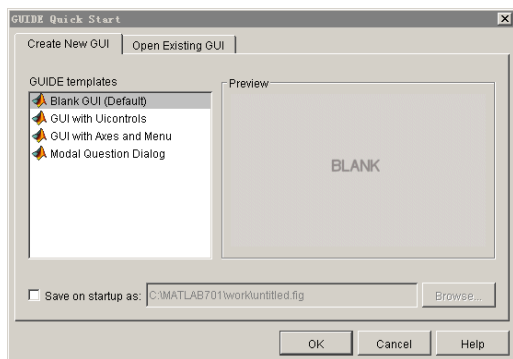


图 6.1 “GUIDE Quick Start”对话框

当用户选择不同的 GUI 模板时，在 GUI 设计窗口会显示出与模板对应的 GUI 图形。一般情况下用户新建的 GUI 文件为空白文件。在对话框中选择“Save on startup as”复选框，可设置新生成的 GUI 文件的保存路径，当然也可以在对 GUI 文件编辑后进行保存。单击“OK”按钮，创建空白的 GUI 文件，弹出如图 6.2 所示的 GUI 设计窗口。新建的 GUI 文件包含一个 fig 文件和包含了 fig 中各控件相关回调函数的 M 脚本。

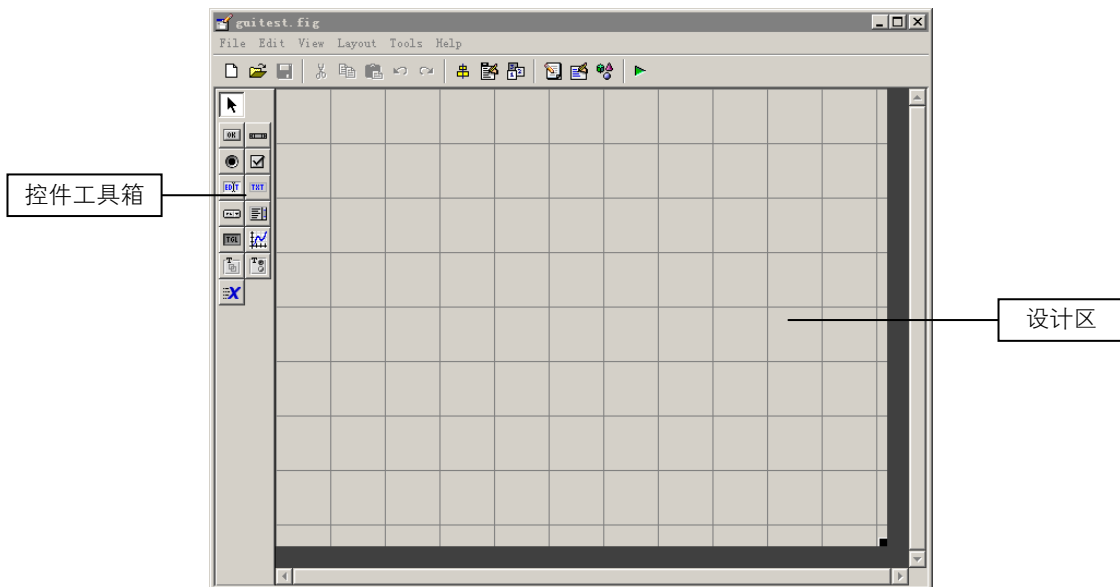


图 6.2 GUI 设计窗口

GUI 设计窗口包含菜单栏、工具栏、控件工具箱、GUI 设计区等，下面简单介绍各部分的功能。

1. 菜单栏

GUI 设计窗口的菜单栏与 MATLAB 一般的界面中的菜单项类似,包括“File”、“Edit”、“View”、“Layout”、“Tools”和“Help”6 个菜单项。其中“File”、“Edit”、“Help”菜单项与其他界面中的功能类似,这里不再详细展开叙述。

1) “View”菜单项

“View”菜单项用于查看 GUI 设计的相关元素,如图 6.3 所示。选择“View”→“Property Inspector”命令,打开对象属性查看器,可查看 GUI 对象的属性。选择“View”→“Object Browser”命令,打开对象浏览器,可查看 GUI 中的基本对象。选择“View”→“M-file Editor”命令,打开 M 文件对象属性查看器,进入 GUI 程序的 M 文件部分。选择“View”→“View Callbacks”中的各子菜单,可以进入 GUI 程序的 M 文件的相应子函数中,查看并编辑。

2) “Layout”菜单项

“Layout”菜单项主要用于控制 GUI 设计窗口中叠放控件的显示顺序,如图 6.4 所示。在创建对象的时候先创建的对象位于下方,如果在其位置再创建一个控件,即为遮盖之前的控件。“Layout”菜单项用于改变控件对象的顺序。同时 Snap to Grid 菜单项用于控制 GUI 控件是否与最近的格线对齐。

3) “Tools”菜单项

“Tools”菜单项主要用于调用 GUI 设计的常用编辑器和相关参数的设置,如图 6.5 所示。其中,选择“Tools”→“Run”命令,执行 GUI 文件;选择“Tools”→“Alignment Objects”命令,可打开控件布局编辑器。选择“Tools”→“Grid and Rulers”命令,可打开网格标尺编辑器。选择“Tools”→“Menu Editor”命令,可打开菜单编辑器。选择“Tools”→“Tab Order Editor”命令,可打开 Tab 顺序编辑器。选择“Tools”→“GUI Options”命令,可打开 GUI 属性设置对话框。

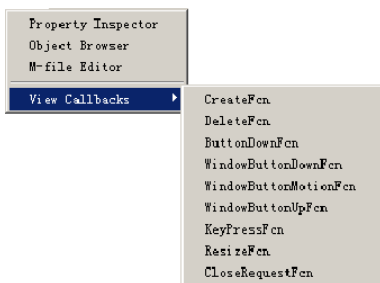


图 6.3 “View”菜单项

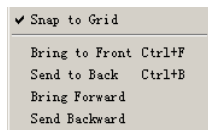


图 6.4 “Layout”菜单项

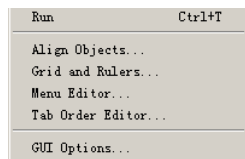









图 6.5 “Tools”菜单项

2. 工具栏

GUI 设计窗口中位于菜单栏下方的工具栏除了具有与 MATLAB 其他窗口中类似的实现常用文件操作功能的图标外,还包括一些可以方便调用 GUI 设计工具和实现有关操作的图标。













- 图标 : 用于打开布局编辑器。
- 图标 : 用于打开菜单编辑器。
- 图标 : 用于打开 Tab 顺序编辑器。
- 图标 : 用于打开 M 文件编辑器。
- 图标 : 用于打开属性查看器。
- 图标 : 用于打开对象浏览器。

- 图标 ：用于执行 GUI 文件。

通过标题栏中的各图标可以方便地打开 GUI 对象的各种编辑工具，在后续的章节将会对这些编辑工具的使用做详细的介绍。

3. 控件工具箱

在 GUI 设计窗口左边的是控件工具箱，包括静态文本框 (Static Text)、可编辑的文本框 (Edit Text)、列表框 (Listbox)、滚动条 (Slider)、按钮 (Push Button)、开关按钮 (Toggle Button)、单选按钮 (Radio Button)、按钮组 (Button Group)、复选框 (Check Box)、列表框 (Listbox)、弹出式菜单 (Popup Menu)、坐标轴 (Axes)、面板 (Panel) 等控件对象，它们是构成 GUI 的基本元素。其中，

- 静态文本框 (Static Text) 控件：对应控件工具箱中的图标 ，用于在界面中显示说明类的文字，用户不可以修改。
- 可编辑的文本框 (Edit Text) 控件：对应控件工具箱中的图标 ，用于用户输入文本，作为 GUI 程序的输入。
- 滚动条 (Slider) 控件：对应控件工具箱中的图标 ，可通过滚动条的滚动，在滚动条相应的位置获取对应的输入参数。
- 按钮 (Push Button) 控件：对应控件工具箱中的图标 ，通过单击它可控制某种程序的执行。
- 开关按钮 (Toggle Button) 控件：对应控件工具箱中的图标 ，用于控制程序的状态，第一次按下按钮，按钮状态为开，第二次按下按钮，按钮状态为关。
- 单选按钮 (Radio Button) 控件：对应控件工具箱中的图标 ，用于在一个选项组内选定一个选项。
- 按钮组 (Button Group) 控件：对应控件工具箱中的图标 ，用于同时选中多个选项。
- 复选框 (Check Box) 控件：对应控件工具箱中的图标 ，用于同时选中多个选项。
- 列表框 (Listbox) 控件：对应控件工具箱中的图标 ，用于显示多个选项。
- 弹出式菜单 (Popup Menu) 控件：对应控件工具箱中的图标 ，在执行一定的鼠标操作后，在界面中显示弹出式菜单。
- 坐标轴 (Axes) 控件：对应控件工具箱中的图标 ，用于在其中绘制图像。
- 面板 (Panel) 控件：对应控件工具箱中的图标 ，便于更好地管理界面上的控件，在面板中的控件与面板的位置可以保持相对不变。

4. GUI 设计区

GUI 设计区用于放置控件和菜单对象。在 GUI 设计的过程中，首先通过鼠标单击选择相应的控件对象，然后在 GUI 设计区按照 GUI 设计的要求，在相应的位置安放 GUI 对象，即可完成界面设计工作。在 GUI 设计区含有网格线，是为了便于用户在合适的地方安置控件对象，而在设计好的 GUI 运行的时候将不再显示网格。

6.2.2 常用控件的设计

控件工具箱内的控件通过鼠标拖曳到 GUI 设计区的相应位置即可。一般为使控件达到 GUI 设计的要求，还需要设置控件的一些属性。通过双击控件对象，可以打开对象属性查看器，对控件相应的属性设置。各控件的公共属性主要有如下几点：

1. 控件风格和外观属性

- BackgroundColor 属性：设置控件背景颜色。

- CData 属性：在控件上显示的真彩色图像。
- ForegroundColor 属性：设置控件前景色。
- String 属性：控件上显示的字符串信息。
- Visible 属性：控件是否可见的控制。

2. 控件的常规信息属性

- Enable 属性：设置控件的状态，“on”表示可选，“off”表示不可选。
- Style：控件对象的类型。
- Tag：控件的标签，用于标识控件，一般由用户使用具有一定意义的词汇命名。
- TooltipString 属性：设置鼠标指针停在此控件上时，显示的提示信息。
- UserData：用户指定数据。
- Position：设置控件的尺寸和位置。
- Units：设置控件的尺寸和位置的单位。
- Font Size：设置控件上显示的字体大小。
- FontName：设置控件上显示的字体类型。
- Value：控件的当前值，回调函数中可根据控件的取值，确定执行的操作。
- Max 和 Min：控制显示的行数，Max-Min>1 为多行；Max-Min≤1 为单行。

3. 控件回调函数的相关属性

- BusyAction：设置处理回调函数的中断。属性值为 Cancel 表示取消中断事件，属性值为 queue 表示排队（默认设置）。
- ButtonDownFcn 属性：在按钮按下时执行的回调函数。
- Callback 属性：设置回调函数的属性，在该对象被选中或改变时，系统将自动响应不同的事件。
- CreateFcn：在对象产生过程中执行的回调函数。
- DeleteFcn：在删除对象过程中执行的回调函数。

各控件的其他一些属性，读者可以在使用的过程中摸索，查阅相关的帮助文档，体会学习 GUI 编程的乐趣。

6.2.3 界面设计窗口的常用工具

在 GUI 设计窗口中，提供了一些工具用于方便地设计 GUI，包括对象属性查看器（Property Inspector）、布局编辑器（Alignment Tool）、对象浏览器（Object Browser）、菜单编辑器（Menu Editor）、网格标尺设置编辑器（Grid and Rulers）和 Tab 顺序编辑器（Tab order edit）。在新建的 GUI 文件窗口中，并没有显示这些工具，需要通过一定的操作打开，在前面的章节中已介绍了各工具的打开方式。下面详细介绍其具体的使用和功能。

1. 对象属性查看器

对象查看器可用于查看并设置 GUI 中各对象的属性，如图 6.6 所示。在对象属性查看器的左侧区域为对象的属性名称，右侧区域显示相应属性的当前属性值。如果为可选属性值的属性，则可以通过单击属性值前的下三角符号▼，从中选择属性值；对于一些比较复杂的属性，则需要通过打开新的窗口设置，例如 BackgroundColor 属性的设置，单击属性值前的图标，将打开如图 6.7 所示的颜色设置对话框设置颜色属性；而其他一些属性，则可以直接修改属性值，输入新的值。

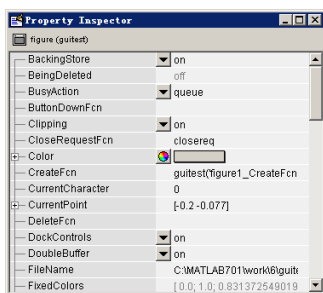


图 6.6 对象属性查看器












图 6.7 颜色设置对话框

2. 菜单编辑器

菜单编辑器可用于创建、设计、修改下拉式菜单和弹出式菜单，如图 6.8 所示。菜单编辑器分为 Menu Bar 和 Context Menus 两个选项卡，分别用于设置下拉式菜单和弹出式菜单。

工具栏可用于对菜单的新建、编辑菜单项，其中，

- 图标 ：用于新建菜单，如果为 Menu Bar 选项卡，则新建下拉式菜单；如果为 Menu Bar 标签页，则新建弹出式菜单。
- 图标 ：用于新建下拉式菜单的菜单项。
- 图标 ：用于新建弹出式菜单项。
- 图标 ：用于对选中的菜单项向上一级菜单移动。
- 图标 ：用于对选中的菜单项向下一级菜单移动。
- 图标 ：用于向上移动选中的菜单项。
- 图标 ：用于向下移动选中的菜单项。
- 图标 ：用于删除选中的菜单项。

下拉式菜单和弹出式菜单的编辑方式类似，现以下拉式菜单为主要对象，讲述菜单编辑器的使用。在“Menu Bar”选项卡下，单击图标  新建下拉式菜单，在菜单编辑器的右侧将显示对菜单项各属性的设置，如图 6.9 所示。

其中“Label”属性为菜单项显示的字符串；“Tag”属性为菜单项的标签。“Accelerator”属性用于设置菜单项的快捷方式，在 Ctrl+后面的输入框输入相应的字母，即构成了该菜单项的快捷方式。选择“Separator above this item”复选框，可在选中的菜单项上添加一条分隔线。选择“Check mark this item”复选框，在第一次访问该菜单项后，会在该菜单项后进行标记。选择“Enable this item”复选框，在第一次打开该菜单项时可用。Callback 属性用于设置菜单项的回调函数，单击其后的“View”按钮，可以查看回调函数；单击“More options”按钮，可以设置更多的菜单项属性。

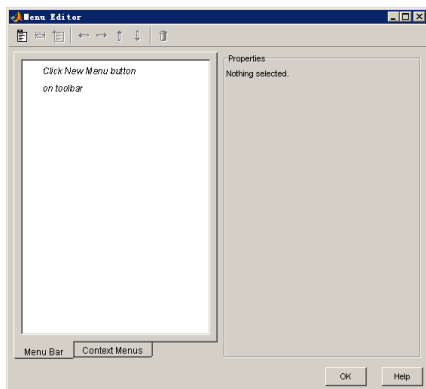


图 6.8 菜单编辑器

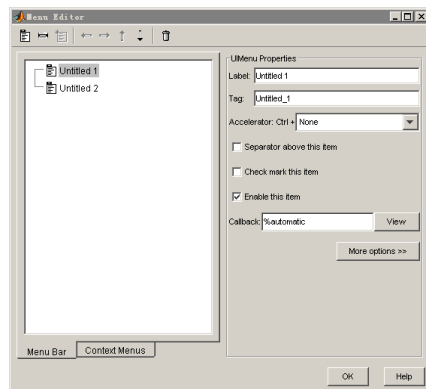
















图 6.9 菜单项的编辑

3. 布局编辑器

布局编辑器可用于调整控件在界面中的水平和垂直方向的布局,如图 6.10 所示,可以设置对齐方式和控件间的间隔。布局编辑器分为 Vertical 和 Horizontal 两个选项区域。

Vertical 选项区域用于设置垂直方向的控件布局,对齐方式 (Align) 设置按钮中,图标  用于关闭垂直设置按钮,图标    分别对应垂直向上对齐、垂直居中对齐、垂直向下对齐;垂直间隔 (Distribute) 设置中,图标    分别对应控件底到顶间隔相等、顶到顶间隔相等、中心到中心间隔相等和底到底间隔相等。同时,在间隔设置过程中,选择“Set spacing”复选框,可以设定间距值,单位为像素。

Horizontal 选项区域用于设置水平方向的控件布局,对齐方式 (Align) 设置按钮中,图标  用于关闭水平设置按钮,图标    分别对应左对齐、居中对齐、右对齐;水平间隔 (Distribute) 设置中,图标    分别对应控件右边到左边间隔相等、左边到左边间隔相等、中心到中心间隔相等和右边到右边间隔相等。同时,在间隔设置过程中,选择“Set spacing”复选框,可以设定水平间距值,单位为像素。

4. 对象浏览器

对象浏览器用于显示当前设计界面中的对象,如图 6.11 所示。对象浏览器可以方便地帮助我们查找对象,单击指定对象,即可选中界面中的该对象,双击指定对象,即可打开对象的属性查看器。

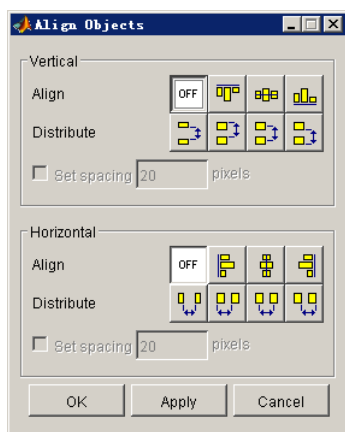


图 6.10 布局编辑器

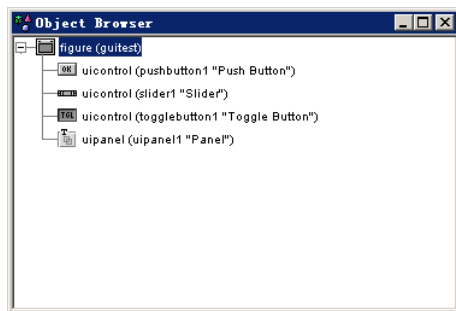




图 6.11 对象浏览器

5. Tab 顺序编辑器

Tab 顺序编辑器用于控制对象的叠放层次,如图 6.12 所示。其中图标  用于控制选中的对象向上移动,而图标  控制选中的对象向下移动。

6. 网格标尺设置编辑器

通过网格标尺设置编辑器,可以在 GUI 设计区添加网格及标尺,方便用户的界面设计,如图 6.13 所示,其中“Show rules”复选框用于控制标尺的显示;“Show guides”复选框用于控制向导线的显示;“Show grid”复选框用于控制 GUI 设计区网格线的显示,同时在“Grid Size(in Pixels)”选框中设置网格线间的间隔;“Snap to grid”复选框用于控制拖曳进来的 GUI 对象是否与最近的格线对齐。

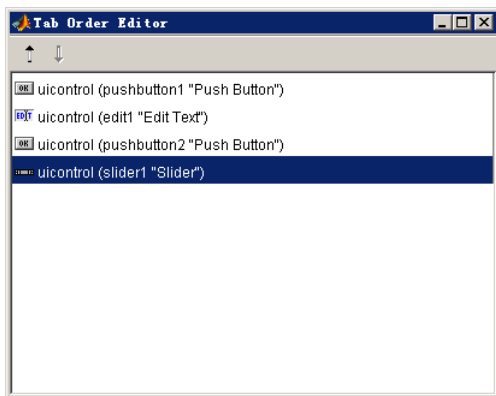


图 6.12 Tab 顺序编辑器

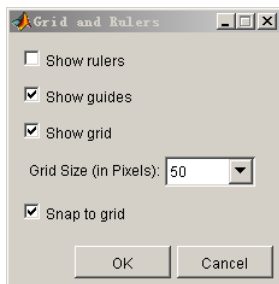


图 6.13 网格标尺设置编辑器

7. GUI 属性设置

“GUI Options” 对话框用于设置 GUI 设计时的属性，如图 6-14 所示，其中：

- Resize behavior 选项用于设置 GUI 界面缩放形式，可选项包括 Non-resizable (固定界面)、Proportional (比例缩放)、User-specified (用户自定义形式)。
- Command-line accessibility 选项用于设置 GUI 对命令窗口的句柄操作的响应方式，可选项包括 Off (完全不响应)、On (响应)、User specified (用户自定义)。

另外，还可以设置 GUI 的保存形式，以单独的图形文件 (.fig) 保存，或者以图形文件和 M 文件 (.fig 和 .M) 保存，并且可以对双文件保存做进一步设置。其中保存的 fig 格式的图形文件包含 GUI 窗口的所有对象的属性值，GUI 的执行主要通过执行该文件；而 M 文件主要保存 GUI 设计窗口的代码和回调函数，当 GUI 运行时，先初始化运行 GUI 设计窗口的代码，生成界面，激活相应的对象，即调用回调函数执行相应的函数功能。

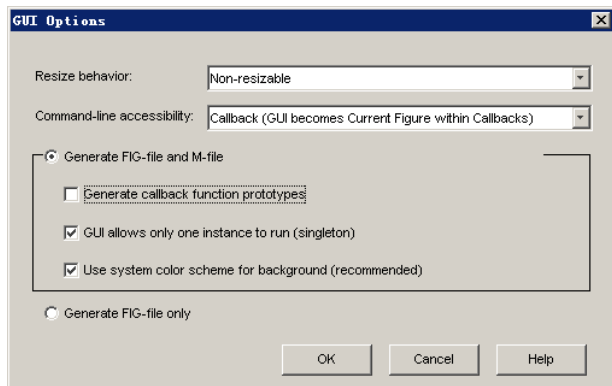


图 6.14 “GUI Options” 对话框

在完成了对 GUI 设计区的控件布局之后，整个图形界面的图形文件的设计基本完成，接下来则是最为重要的实现图形界面对象功能。MATLAB 中的 GUI 对象主要通过事件驱动的形式，利用回调函数的设置响应相应的功能。简单来讲，为设置某一 GUI 对象的回调程序，只要用鼠标右键单击该对象，在弹出的快捷菜单中选择“View Callbacks”菜单项的各子菜单，即可设置一种方式调用回调程序。

只要 GUI 对象指定的事件发生，将自动调用某指定的函数，实现相应的函数功能。关于回调函数的编程相关知识，将在下一节中详细展开叙述。

6.3 利用程序语言设计 GUI

在 6.2 节中主要介绍了 GUI 图形界面中的一些常用的控件对象、菜单对象，如何利用 GUIDE 实现，本节首先简单介绍这些 GUI 对象如何利用编程的方式实现。在完整的图形界面中，常常需要一些对话框来实现重要信息的显示，同时，文件的打开、保存、打印等操作也往往需要依赖于打开 Windows 的相应的对话框来实现。GUI 中对话框的设计主要通过编程实现，这部分内容将在 6.3.2 小节中介绍。

6.3.1 GUI 对象编程

除了使用 GUI 向导设计 GUI 对象外，也可以通过编程的方式实现。但是编程的方式在具体操作上远不如 GUI 向导简单、直观，所以本小节仅简单介绍下 GUI 对象编程技术，感兴趣的读者可以进一步阅读相关的文档。

1. 图形窗口的设计

图形窗口的创建函数为 `figure()`，其调用格式如下。

- `figure`：使用默认的属性值创建新的图形窗口，并使该图形窗口为当前窗口。
- `figure('PropertyName',PropertyValue,...)`：创建图形窗口，并设置窗口属性指定的属性值。
- `figure(h)`：创建句柄为 `h` 的图形窗口，如果句柄为 `h` 的图形窗口已存在，即使句柄为 `h` 窗口为当前窗口。
- `h = figure(...)`：创建图形窗口，并返回句柄值 `h`。

其中，在创建图形窗口时，往往需要根据 GUI 设计的要求，设置界面的相应属性。在图形窗口设计中常用到的属性如下。

- `MenuBar` 属性：控制窗口的菜单栏和标题栏的显示。在默认情况下生成的窗口包含菜单栏和标题栏，当 `MenuBar` 属性的属性值设置为“none”时生成的窗口不包含菜单栏和标题栏。
- `NumberTitle` 属性：控制窗口数字标题的显示，即 Figure 1、Figure 2、Figure 3 等，默认情况下新生成的窗口会按顺序显示数字标题，如果设置 `NumberTitle` 属性的属性值为“off”即可不显示数字标题。
- `Name` 属性：设置窗口的标题，默认情况下紧跟在数字标题后显示。
- `Tag` 属性：图形窗口的标识，在程序设计中识别不同的窗口。
- `Position` 属性：设置生成的图形窗口的大小。
- `Units` 属性：设置图形窗口大小的单位。
- `KeyPressFcn` 属性：设置键盘键按下响应事件。
- `WindowButtonDownFcn` 属性：设置鼠标键按下响应事件。
- `WindowButtonMotionFcn` 属性：设置鼠标移动响应事件。
- `WindowButtonUpFcn` 属性：设置鼠标键释放响应事件。

【例 6.1】图形窗口的设计。

```
%创建标题栏仅显示 gui figure design,无菜单栏和标题栏且固定大小的窗口
h=figure('name','gui figure design','menubar','none',...
    'numbertitle','off','position',[100 300 500 100]);
```

执行上述程序，将生成如图 6.15 所示的图形窗口。

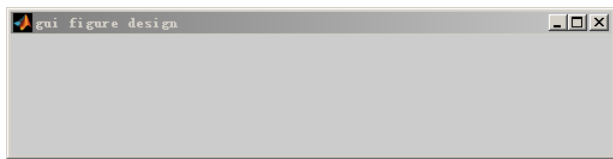


图 6.15 图形窗口的设计

2. 控件的设计

函数 `uicontrol()` 可用于控件的设计，其调用格式如下。

- `handle = uicontrol('PropertyName',PropertyValue,...)`: 创建控件，并设置指定属性的属性值。
- `handle = uicontrol(parent,'PropertyName',PropertyValue,...)`: 在 `parent` 句柄指定的控件中创建控件，可以作为控件容器的控件主要有图形窗口、面板和按钮组。
- `handle = uicontrol`: 以默认的设置创建控件并返回句柄 `handle`。
- `uicontrol(uich)`: 输入焦点设置到句柄 `uich` 指定的对象上。

控件设计的基本属性在前面章节中已有介绍，且不同控件间属性的差异也不大，这里主要详细介绍其中的 `style` 属性，该属性用于指定不同的控件对象，其中属性值设置为 `checkbox`（创建复选框对象）、`listbox`（创建列表框对象）、`popupmenu`（创建下拉式菜单对象）、`pushbutton`（创建命令按钮对象）、`radiobutton`（创建单选按钮对象）、`slider`（创建滚动条对象）、`text`（创建静态文本框对象）、`togglebutton`（创建开关按钮对象）。

【例 6.2】控件的设计。

```
h=figure('name','GUI 控件设计','menubar','none')           %生成 GUI 窗口
h_text=uicontrol(h,'style','text',...                      %创建静态文本框
    'unit','normalized','position',[0.3,0.5,0.25,0.14],...
    'horizontal','left','string',{'GUI 控件设计'},'FontSize',18);
set(h_text,'backgroundcolor','g');                         %设置静态文本框的属性
```

执行上述程序，将生成如图 6.16 所示的窗口。

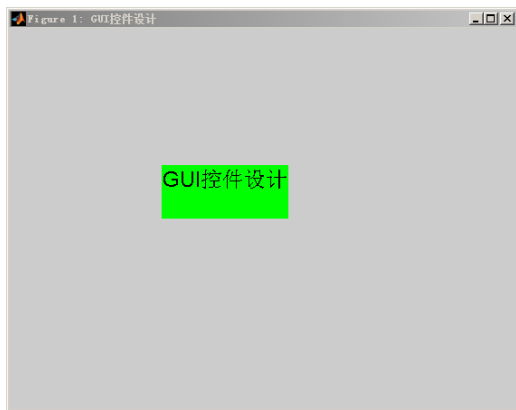


图 6.16 控件的设计

3. 坐标轴的设计

建立坐标轴对象使用函数 `axes()`，其调用格式如下。

- `axes`: 创建默认状态下的坐标轴对象。
- `axes('PropertyName',PropertyValue,...)`: 创建坐标轴对象，并指定其属性。
- `axes(h)`: 设置句柄 `h` 的坐标轴对象为当前对象。
- `h = axes(...)`: 创建坐标轴对象，并返回其句柄 `h`。

【例 6.3】坐标轴的设计。

```
figure; %创建新的图形窗口
h=axes('xlim',[0 15],'ylim',[0 1.8],'fontsize',12); %设置坐标轴
```

执行上述程序，将生成如图 6.17 所示的含有坐标轴的窗口。

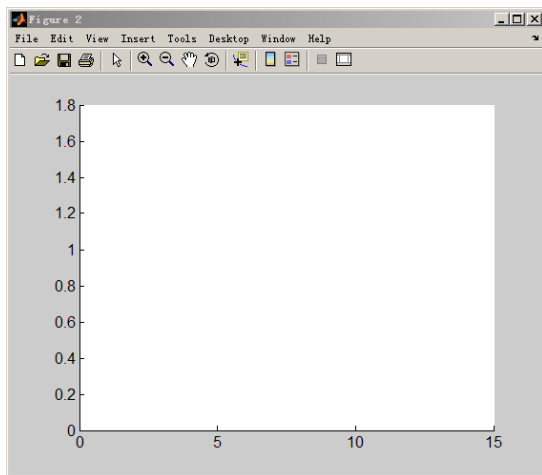


图 6.17 坐标轴的设计

4. 菜单的设计

利用函数 `uimenu()` 和 `uicontextmenu()` 可分别建立下拉式菜单和弹出式菜单，其中，

(1) `uimenu()` 函数用于建立下拉式菜单的一级菜单项和子菜单项，其调用格式如下。

- `uimenu('PropertyName',PropertyValue,...)`: 创建下拉式菜单，并设置相应的属性。
- `uimenu(parent,'PropertyName',PropertyValue,...)`: 创建以句柄 `parent` 指定的菜单为父对象的子菜单。
- `handle = uimenu('PropertyName',PropertyValue,...)`: 创建下拉式菜单，并返回句柄 `handle`。
- `handle = uimenu(parent,'PropertyName',PropertyValue,...)`: 创建下拉式菜单的子菜单，并返回句柄 `handle`。

(2) 弹出式菜单建立的函数为 `uicontextmenu()`，其调用格式如下。

- `handle = uicontextmenu('PropertyName',PropertyValue,...)`: 创建弹出式菜单，设置相应的属性，并返回句柄 `handle`。

在利用函数 `uicontextmenu` 创建了弹出式菜单后，函数 `uimenu` 用于进一步设置弹出式菜单的菜单项，属性 `UIContextMenu` 用于建立弹出式菜单与对象的联系，设置选择弹出式菜单的容器。

【例 6.4】菜单的设计。

```
h_fig=figure;
h_menu=uimenu('Label','&Color'); %创建下拉菜单
h_submenu1=uimenu(h_menu,'Label','Green',... %创建下拉菜单的子菜单
    'Callback','set(h_fig,'color','blue')');
h_submenu2=uimenu(h_menu,'label','Red',...
    'Callback','set(h_fig,'color','red')');
h_submenu2=uimenu(h_menu,'label','Blue',...
    'Callback','set(h_fig,'color','blue')');
h_uimenu=uicontextmenu; %创建弹出式菜单
set(h_fig,'UIContextMenu',h_uimenu); %弹出式菜单与图形界面关联
%设置弹出式菜单的菜单项
uimenu(h_uimenu,'label','Red','callback','set(h_fig,'color','r'),'')
uimenu(h_uimenu,'label','Blue','callback','set(h_fig,'color','b'),'')
uimenu(h_uimenu,'label','Green','callback','set(h_fig,'color','g'),'')
```

执行上述程序，将生成如图 6.18 所示的下拉式菜单和如图 6.19 所示的弹出式菜单。

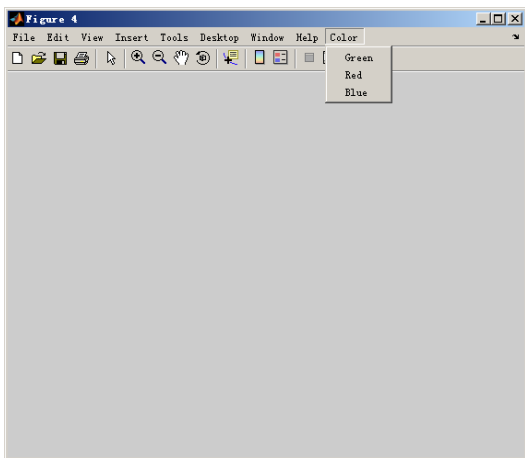


图 6.18 下拉式菜单

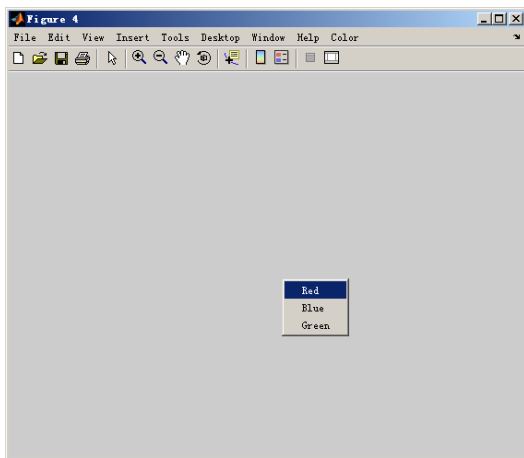


图 6.19 弹出式菜单

6.3.2 GUI 的对话框

GUI 设计中的对话框可以分为两类，一类是公共对话框，即 Windows 系统中的对话框，包括文件打开对话框、文件保存对话框、颜色设置对话框、字体设置对话框、打印预览对话框、打印对话框等；另一类为 MATLAB 专用对话框，即 MATLAB 自带的对话框，包括错误信息对话框、帮助对话框、输入对话框、列表选择对话框、消息对话框、问题提示对话框、警告对话框、进程条、目录对话框。

1. 公共对话框

公共对话框是 Windows 系统的对话框，在 MATLAB 中通过调用相应函数实现对话框的调用。

1) 文件打开对话框

函数 `uigetfile()` 用于打开文件打开对话框，其调用格式如下。

- `uigetfile`: 打开文件打开对话框，对话框中显示了当前路径下的所有文件，可选择打开系统中任意路径下的文件。
- `uigetfile('FilterSpec')`: 打开文件打开对话框，`FilterSpec` 参数用于设置打开的文件类型。
- `uigetfile('FilterSpec','DialogTitle')`: 打开文件打开对话框，`FilterSpec` 参数用于设置打开的文件类型，`DialogTitle` 参数用于设置文件打开对话框的标题。
- `uigetfile('FilterSpec','DialogTitle','DefaultName')`: 打开文件打开对话框，`FilterSpec` 参数用于设置打开的文件类型，`DialogTitle` 参数用于设置文件打开对话框的标题，`DefaultName` 参数用于设置默认打开的文件名。
- `[FileName,PathName] = uigetfile(...)`: 打开文件打开对话框，返回 `FileName` 参数为打开的文件名，`PathName` 参数为打开文件的路径。
- `[FileName,PathName,FilterIndex] = uigetfile(...)`: 打开文件打开对话框，返回 `FileName` 参数为打开的文件名，`PathName` 参数为打开文件的路径，`FilterIndex` 参数为打开类型的索引，从 1 开始，如果取消或者打开文件失败则返回的 `FilterIndex` 参数值为 0。

【例 6.5】“打开 M 文件”对话框。

```
[filename,filepath]=uigetfile('*.*','打开 M 文件');%调用文件打开对话框，打开 M 文件
```

执行上述程序，打开如图 6.20 所示的“打开 M 文件”对话框。

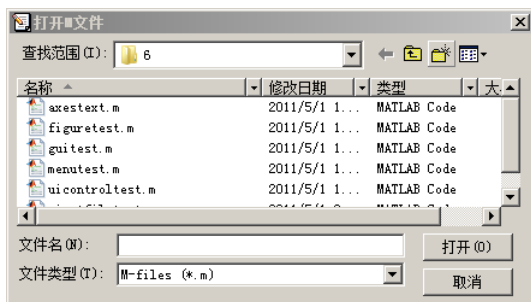


图 6.20 “打开 M 文件”对话框

2) 文件保存对话框

函数 `uiputfile()` 用于打开文件保存对话框，其调用格式如下。

- `uiputfile`: 打开文件保存对话框，对话框中显示当前路径下的文件，输入文件名，保存文件。
- `uiputfile('FilterSpec')`: 打开文件保存对话框，参数 `FilterSpec` 用于设置文件的保存类型。
- `uiputfile('FilterSpec','DialogTitle')`: 打开文件保存对话框，参数 `FilterSpec` 用于设置文件的保存类型，参数 `DialogTitle` 用于设置文件保存对话框的标题。
- `uiputfile('FilterSpec','DialogTitle','DefaultName')`: 打开文件保存对话框，参数 `FilterSpec` 用于设置文件的保存类型，参数 `DialogTitle` 用于设置文件保存对话框的标题，`DefaultName` 参数用于设置默认的文件保存名。
- `[FileName,PathName] = uiputfile(...)`: 打开文件保存对话框，保存文件，返回保存文件的文件名和路径。
- `[FileName,PathName,FilterIndex] = uiputfile(...)`: 打开文件保存对话框，保存文件，返回保存文件的文件名、路径和索引值。

【例 6.6】“保存 M 文件”对话框。

```
[filename,filepath]=uiputfile('*.m','保存 M 文件');%调用文件保存对话框，保存 M 文件
```

执行上述程序，打开如图 6.21 所示的“保存 M 文件”对话框。

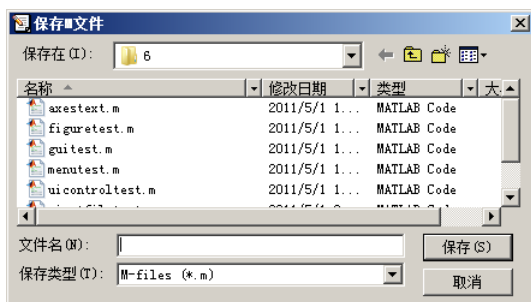


图 6.21 “保存 M 文件”对话框

3) 颜色设置对话框

函数 `uigetcolor()` 可调用颜色设置对话框，用于图形对象颜色的交互设置，其调用格式如下。

`c = uigetcolor(h_or_c, 'DialogTitle')`: 打开颜色设置对话框，其中参数 `h_or_c` 为颜色的初始值，`DialogTitle` 为颜色设置对话框的标题，返回参数为用户选择的颜色。

【例 6.7】“颜色设置”对话框。

```
c = uigetcolor([0 1 1], '颜色设置')%调用颜色设置对话框
```

执行上述程序，生成如图 6.22 所示的“颜色设置”对话框。



图 6.22 “颜色设置”对话框

4) 字体设置对话框

函数 `uifont()` 用于字体属性的交互式设置，可以设置的字体属性包括字体类型、字体大小、字体粗细和字体倾斜度等，其调用格式如下。

- `uifont`: 打开字体设置对话框，设置字体属性。
- `uifont(h)`: 以句柄 `h` 的字体属性为打开的字体设置对话框的初始属性。
- `uifont(S)`: 参数 `S` 为指定的字体属性值，即设置初始打开字体设置对话框时的字体属性。
- `uifont(h,'DialogTitle')`: 打开字体设置对话框，以句柄 `h` 的字体属性为打开的字体设置对话框的初始属性，`DialogTitle` 参数为字体设置对话框的标题。
- `uifont(S,'DialogTitle')`: 打开字体设置对话框，以参数 `S` 中的设置为字体设置对话框的初始属性，`DialogTitle` 参数为字体设置对话框的标题。
- `S = uifont(...)`: 参数 `S` 返回设置的字体属性。

【例 6.8】“字体”对话框。

`uifont`

执行上述程序，生成如图 6.23 所示的“字体”对话框。

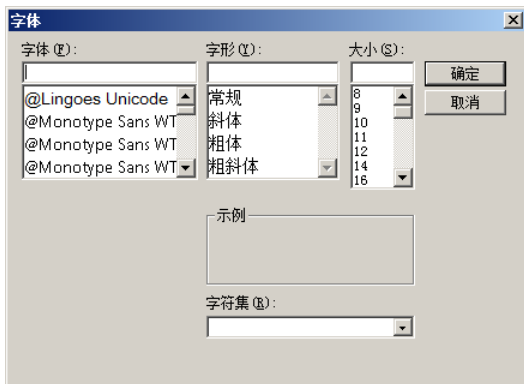


图 6.23 “字体”对话框

5) 页面设置对话框

函数 `pagesetupdlg()` 用于页面设置的交互式设置，其调用格式如下。

`dlg = pagesetupdlg(fig)`: 打开页面设置对话框，`fig` 为需要进行页面设置的图形窗口的句柄。

【例 6.9】页面设置对话框。

```
h=figure;
ezplot('sin(x)');
dlg = pagesetupdlg(h);
```

执行上述程序，生成如图 6.24 所示的页面设置对话框。

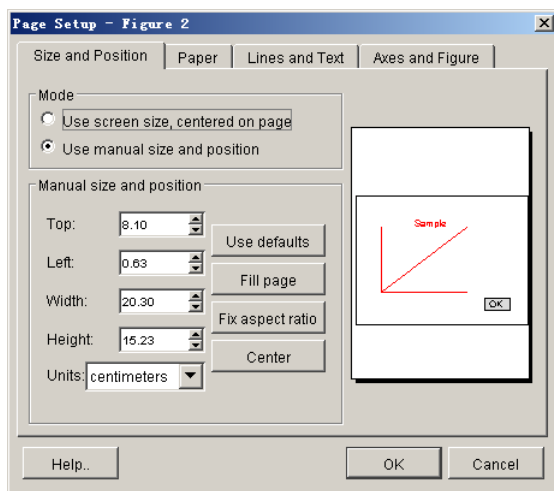


图 6.24 页面设置对话框

6) 打印预览对话框

函数 `printpreview()` 用于对打印页面进行预览，其调用格式如下。

- `printpreview`: 显示当前图形窗口的打印预览效果。
- `printpreview(f)`: 显示句柄 `f` 指定的图形窗口的打印预览效果。

【例 6.10】打印预览窗口。

```
h=figure;
ezplot('sin(x)');
printpreview           %打开打印预览窗口
```

执行上述程序，打开如图 6.25 所示的窗口。

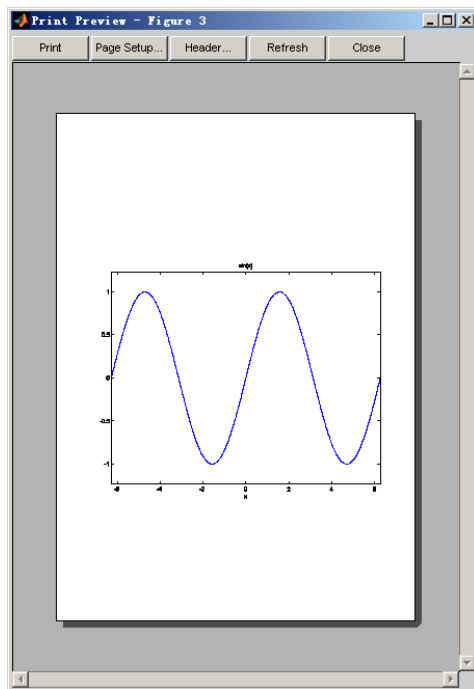


图 6.25 打印预览窗口

7) 打印对话框

函数 `printdlg()` 用于打开打印对话框，其调用格式如下。

- `printdlg`: 打开打印对话框，打印当前的图形窗口。
- `printdlg(fig)`: 打开打印对话框，打印句柄 `fig` 指定的图形窗口。

【例 6.11】“打印”对话框。

```
h=figure;
ezplot('sin(x)');
printdlg      %打开“打印”对话框
```

执行上述程序，打开如图 6.26 所示的“打印”对话框，可对当前图形窗口执行打印操作。

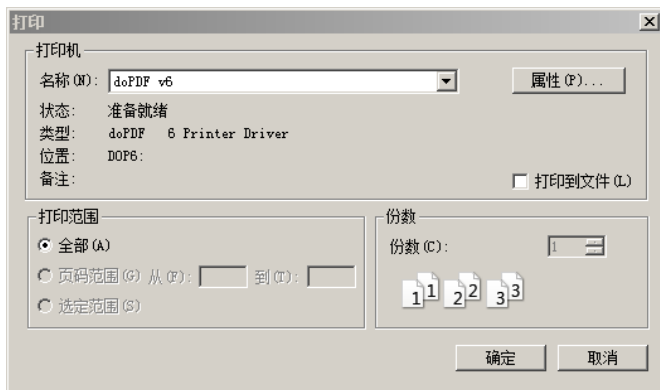


图 6.26 “打印”对话框

2. MATLAB 专用对话框

在 MATLAB 中除了可以打开系统自带的对话框外，MATLAB 也提供了专用的对话框，便于用户与 MATLAB 程序交互使用。下面详细介绍这些对话框。

1) 错误信息对话框

错误信息对话框可用于提示程序运行过程中的错误信息，函数 `errordlg()` 用于打开错误信息对话框，其调用格式如下。

- `errordlg`: 打开默认的错误信息对话框，对话框的标题为“Error Dialog”，显示的错误提示信息为“This is the default error string”，且错误信息对话框中有“OK”按钮，单击此按钮确定错误信息或关闭错误信息对话框。
- `errordlg('errorstring')`: 打开错误信息对话框，显示的错误提示信息为 `errorstring`。
- `errordlg('errorstring','dlgname')`: 打开错误信息对话框，对话框的标题为输入参数 `dlgname`，显示的错误提示信息为 `errorstring`。
- `h = errordlg(...)`: 打开错误信息对话框，并返回其句柄 `h`。

【例 6.12】“错误提示”对话框。

```
errordlg('程序执行错误','错误提示')
```

执行上述程序，生成如图 6.27 所示的“错误提示”对话框。



图 6.27 “错误提示”对话框

2) 帮助对话框

函数 `helpdlg()` 可用于打开帮助对话框，提示帮助信息，其调用格式如下。

- `helpdlg`: 打开默认状态下的帮助对话框，其标题为“Help Dialog”，默认提示的帮助字符串为“This is the default help string”。
- `helpdlg('helpstring')`: 打开标题为“Help Dialog”，提示的帮助信息为“helpstring”的帮助对话框。
- `helpdlg('helpstring','dlgname')`: 打开标题为“dlgname”，提示的帮助信息为“helpstring”的帮助对话框。
- `h = helpdlg(...)`: 打开帮助对话框，并返回其句柄 `h`。

【例 6.13】“帮助提示”对话框。

```
helpdlg('此空格需要输入大于 0 的正数','帮助提示')
```

执行上述程序，生成如图 6.28 所示的“帮助提示”对话框。



图 6.28 “帮助提示”对话框

3) 输入对话框

函数 `inputdlg()` 用于调用输入对话框，输入对话框用于程序运行过程中交互式的获取用户的输入信息，其调用格式如下。

- `answer = inputdlg(prompt)`: 打开输入对话框，参数 `prompt` 为单元数组，包含输入数据的提示，返回参数 `answer` 为通过输入对话框获取用户的输入值。
- `answer = inputdlg(prompt,dlg_title)`: 打开输入对话框，参数 `prompt` 为单元数组，包含输入数据的提示，参数 `dlg_title` 为输入对话框的标题，返回参数 `answer` 为通过输入对话框获取用户的输入值。
- `answer = inputdlg(prompt,dlg_title,num_lines)`: 打开输入对话框，参数 `prompt` 为单元数组，包含输入数据的提示，参数 `dlg_title` 为输入对话框的标题，参数 `num_lines` 为用户输入值的行数，返回参数 `answer` 为通过输入对话框获取用户的输入值。
- `answer = inputdlg(prompt,dlg_title,num_lines,defAns)`: 打开输入对话框，其中参数 `defAns` 指定输入对话框的默认值。

【例 6.14】输入对话框。

```
prompt={'请输入矩阵的行数','请输入矩阵的列数'};
dlg_title='matrix';
num_lines=1;                                     %输入对话框的行数
default_val={'3','3'};                             %默认的值
answer=inputdlg(prompt,dlg_title,num_lines,default_val); %打开输入对话框
A=rand(str2num(answer{1}),str2num(answer{2}))
```

执行上述程序，打开如图 6.29 所示的输入对话框。

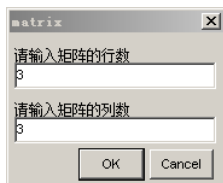


图 6.29 输入对话框

4) 列表选择对话框

函数 `listdlg()` 可用于生成列表选择对话框, 用于在多个选项中选择需要的值, 其调用格式如下。

- `[Selection,ok] = listdlg('ListString',S,...)`: 打开列表选择对话框, 输入参数 `S` 指定列表框的属性 `ListString` (列表框选择的字符串), 函数返回的参数 `Selection` 为用户从列表框中选择的字符串的下标, `OK` 值用于确定用户是否确定这次选择, 用于确定, 单击“OK”按钮, `OK` 的值为 1, 否则为 0。同时列表选择对话框还有其他一些属性可以设置, 它们是 `SelectionMode` 属性 (设置列表的选择模式, 可设为 `'single'` 或者 `'multiple'`, 其中 `'multiple'` 为默认设置)、`ListSize` 属性 (设置列表选择对话框的大小)、`InitialValue` 属性 (设置列表选择对话框初始选中的选项, 默认情况下为第一项)、`Name` 属性 (列表选择对话框的标题)、`PromptString` 属性 (设置列表选择对话框的提示字符串)、`OKString` 属性 (列表选择对话框中 `OK` 按钮上的字符串, 默认为“OK”)、`CancelString` 属性 (列表选择对话框中 `Cancel` 按钮上的字符串, 默认为“Cancel”)、`uh` 属性 (按钮的高度)、`fus` (按钮框架的高度) 和 `ffs` 属性 (列表选择对话框的边界设置)。另外, 列表选择对话框中的“Select All”按钮可用于选择所有的选项。

【例 6.15】列表选择对话框。

```
S={'赤','橙','红','绿','青','蓝','紫'}
[Selection,ok] = listdlg('ListString',S,'PromptString','选择一种颜色')
执行上述程序, 生成如图 6.30 所示的列表选择对话框。
```

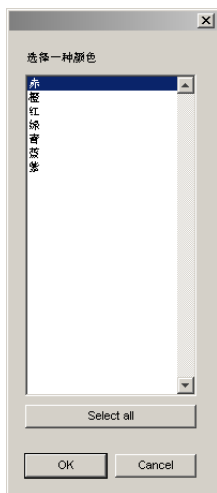


图 6.30 列表选择对话框

5) 消息对话框

函数 `msgbox()` 用于打开消息对话框, 显示提示信息, 其调用格式如下。

- `msgbox(message)`: 打开消息对话框, 参数 `message` 为消息对话框中提示的消息。
- `msgbox(message,title)`: 打开消息对话框, 参数 `message` 为消息对话框中提示的消息, 参数 `title` 为消息对话框的标题。
- `msgbox(message,title,'icon')`: 打开消息对话框, 其中输入参数 `icon` 用于设置消息对话框显示的图标, 可以设置的值为 `none`、`error`、`help`、`warn`、`custom`, 默认状态下为 `none`。
- `msgbox(message,title,'custom',iconData,iconCmap)`: 打开消息对话框, 并设置自定义的图标, 参数 `iconData` 设置图标的数据, `iconCmap` 是图标的颜色。

- `h = msgbox(...)`: 打开消息对话框, 并返回句柄 `h`。

【例 6.16】“错误”对话框。

```
msgbox('运行出错','错误','error')
```

执行上述程序, 显示如图 6.29 所示的“错误”对话框。

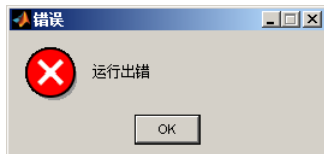


图 6.31 “错误”对话框

6) 问题提示对话框

函数 `questdlg()` 用于打开问题提示对话框, 用于与用户交互判断问题, 其调用格式如下。

- `button = questdlg('qstring')`: 打开问题提示对话框, 并提问 `qstring`, 默认情况下对话框中有 Yes、No、Cancel 3 个按钮, 返回参数 `button` 用于记录用户按下的按钮。
- `button = questdlg('qstring','title')`: 打开问题提示对话框, 参数 `title` 用于设置对话框的标题。
- `button = questdlg('qstring','title','default')`: 打开问题提示对话框, 参数 `default` 用于设置初始打开时的激活按钮, 即按下回车键响应的按钮。
- `button = questdlg('qstring','title','str1','str2','default')`: 打开问题提示对话框, 有两个按钮, 分别显示 `str1` 和 `str2`。
- `button = questdlg('qstring','title','str1','str2','str3','default')`: 打开问题提示对话框, 有 3 个按钮, 分别显示 `str1`、`str2` 和 `str3`。

【例 6.17】问题提示对话框。

```
button = questdlg('Are you sure')
```

执行上述程序, 生成如图 6.32 所示的对话框。



图 6.32 问题提示对话框

7) 警告对话框

函数 `warndlg()` 用于打开警告对话框, 其调用格式如下。

`h = warndlg('warningstring','dlgname')`: 打开警告对话框, 参数 `warningstring` 为警告的内容, 参数 `dlgname` 为警告对话框的标题, 返回对话框的句柄 `h`。

【例 6.18】警告对话框。

```
h = warndlg('输入参数需大于0','警告对话框')
```

执行上述程序, 生成如图 6.33 所示的对话框。



图 6.33 警告对话框

8) 进程条

函数 `waitbar()` 生成进程条, 以图形化的方式显示运算或处理的进程, 其调用格式如下。

- `h = waitbar(x,'title')`: 显示进程条, 显示程序处理的进程, 其中输入参数 `x` 需为 $[0,1]$ 的数, 控制进程条的显示比例, `title` 为进程条的标题, 参数 `h` 返回进程条的句柄。
- `waitbar(...,property_name,property_value,...)`: 显示进程条, 并设置进程条相应的属性。
- `waitbar(x)`: 显示进程条, 进程条设置到 `x` 的比例处。
- `waitbar(x,h)`: 显示进程条, 句柄 `h` 的进程条设置到 `x` 的比例处。

【例 6.19】进程条。

```
h = waitbar(0,'Please wait...')
for i=1:100
    x=i/100;
    waitbar(x,h,['程序已运行了' num2str(x*100)]);
    drawnow;
end
close(h)
```

执行上述程序, 在程序运行过程中将显示如图 6.34 所示的进程条。

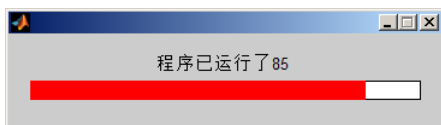


图 6.34 进程条

9) 目录对话框

函数 `uigetdir()` 用于生成目录对话框, 其调用格式如下。

- `directory_name = uigetdir`: 打开目录对话框, 浏览目录, 参数 `directory_name` 返回选择的目录。
- `directory_name = uigetdir('start_path')`: 打开目录对话框, 显示目录 `start_path`, 参数 `directory_name` 返回选择的目录。
- `directory_name = uigetdir('start_path','dialog_title')`: 打开目录对话框, 参数 `dialog_title` 设置对话框的标题。

【例 6.20】目录对话框。

```
directory_name = uigetdir
```

执行上述程序, 生成如图 6.35 所示的目录对话框, 选择目录。



图 6.35 目录对话框

6.4 GUI 文件

在通常情况下，一个 GUI 文件包括两个文件，即 fig 文件和 M 文件，其中 fig 主要包括界面控件的信息，而 M 文件中包括了初始化生成界面的语句和实现 GUI 功能的回调函数。本章将首先介绍生成的 GUI 的 M 文件的结构，方便用户直接在 M 文件中查看、编辑 GUI。

在前面的章节中，介绍了如何利用 GUI 向导或者编程方式创建 GUI 界面中的控件等，但是 GUI 功能的实现需要依靠回调函数来实现，本章中将详细介绍回调函数及其参数传递的相关知识。

6.4.1 M 文件结构

在利用 GUI 向导完成了界面设计后，向控件添加回调函数时，会自动生成 M 文件，其中包括了之前对界面对象的布局、外观属性进行设置，本小节将向用户介绍如何读懂这些 M 文件。通常 GUI 设计中的 M 文件主要由以下几个部分组成：注释、初始化代码、Opening 函数、Output 函数和回调函数。其中：

- 初始化代码为用户在利用 GUI 向导设计 GUI 时生成的 M 文件，包括对控件、菜单对象布局、属性的设置，窗口相关属性的设置等。
- Opening 函数为 GUI 所有控件创建完成，在 GUI 显示之前执行的，通过 Opening 函数用户可以设置一些需要程序初始化时完成的任务，通常 Opening 函数的文件名为“GUI 文件名_OpeningFcn”。
- Output 函数用于向命令行返回输出结果，输出参数需要在 Output 函数中生成，Output 函数的文件名为“GUI 文件名_OutputFcn”。
- 回调函数为 GUI 对象接收到一定的操作后，执行的指定的函数，回调函数的输入参数为 hObject、eventdata、handles，回调函数名为“控件标签名_回调函数属性名”。

下面以一个实例，向读者具体展示 M 文件各部分组成。

【例 6.21】GUI M 文件的阅读。

以下为利用 GUI 向导设计界面后，编写回调函数前自动生成的 M 文件。

```
function varargout = guitest1(varargin)
%以下部分为程序的注释
% GUITEST1 M-file for guitest1.fig
%   GUITEST1, by itself, creates a new GUITEST1 or raises the existing
%   singleton*.
%   H = GUITEST1 returns the handle to a new GUITEST1 or the handle to
%   the existing singleton*.
%   GUITEST1('CALLBACK',hObject,eventData,handles,...) calls the local
%   function named CALLBACK in GUITEST1.M with the given input arguments.
%   GUITEST1('Property','Value',...) creates a new GUITEST1 or raises the
%   existing singleton*. Starting from the left, property value pairs are
%   applied to the GUI before guitest1_OpeningFunction gets called. An
%   unrecognized property name or invalid value makes property application
%   stop. All inputs are passed to guitest1_OpeningFcn via varargin.
%   *See GUI Options on GUIDE's Tools menu. Choose "GUI allows only one
%   instance to run (singleton)".
% See also: GUIDE, GUIDATA, GUIHANDLES
% Copyright 2002-2003 The MathWorks, Inc.
% Edit the above text to modify the response to help guitest1
% Last Modified by GUIDE v2.5 02-May-2011 15:37:11
% Begin initialization code - DO NOT EDIT
```

```

% 以下部分为界面的初始化
gui_Singleton = 1;
gui_State = struct('gui_Name',       mfilename, ...
                  'gui_Singleton',   gui_Singleton, ...
                  'gui_OpeningFcn', @guitest1_OpeningFcn, ...
                  'gui_OutputFcn',  @guitest1_OutputFcn, ...
                  'gui_LayoutFcn',   [] , ...
                  'gui_Callback',    []);
if nargin && ischar(varargin{1})
    gui_State.gui_Callback = str2func(varargin{1});
end
if nargout
    [varargout{1:nargout}] = gui_mainfcn(gui_State, varargin{:});
else
    gui_mainfcn(gui_State, varargin{:});
end
% End initialization code - DO NOT EDIT
%GUI 界面初始化工作完成
%以下为用户访问 GUI 之前的初始化任务
% --- Executes just before guitest1 is made visible.
function guitest1_OpeningFcn(hObject, eventdata, handles, varargin)
% This function has no output args, see OutputFcn.
% hObject    handle to figure 当前对象句柄
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA) 句柄的数据结构
% varargin    command line arguments to guitest1 (see VARARGIN) 可变的输入参数
% Choose default command line output for guitest1
handles.output = hObject;
% Update handles structure 更新
guidata(hObject, handles);%保存 handles
% UIWAIT makes guitest1 wait for user response (see UIRESUME)
% uiwait(handles.figure1);
%以下部分将输出参数返回到命令行
% --- Outputs from this function are returned to the command line.
function varargout = guitest1_OutputFcn(hObject, eventdata, handles)
% varargout  cell array for returning output args (see VARARGOUT);
% hObject    handle to figure
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)
% Get default command line output from handles structure
varargout{1} = handles.output;%从句柄中获取数据输出
%以下部分为回调函数
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles     structure with handles and user data (see GUIDATA)

```

6.4.2 回调函数

本节主要介绍实现控件功能的回调函数。用户通过一定的操作（例如单击某个按钮等），激活当前的 GUI 对象，执行相应的回调函数，并返回计算结果。

1. 回调函数的类型

回调函数的类型与 GUI 对象的回调函数属性相对应，而不同的对象拥有的回调函数属性也不同，每一种回调函数对应不同的触发机制。下面以回调函数最为常用的 GUI 对象按钮和窗口为例，介绍不同的回调函数类型，其类型基本涵盖了其他对象的回调函数属性。

（1）按钮的回调函数属性如下。

- Callback 属性：用户单击按钮时执行回调函数。
- CreateFcn 属性：初始化显示该按钮时执行回调函数。
- ButtonDownFcn 属性：在按钮上或其周围 5 个像素范围内单击鼠标时执行回调函数。
- DeleteFcn 属性：在删除按钮过程中执行回调函数。
- KeyPressFcn 属性：在按钮上或键盘按下任意键执行回调函数。

(2) 图形窗口的回调函数属性如下。

- KeyPressFcn 属性：在图形窗口上按下键盘上任意键执行回调函数。
- WindowButtonDownFcn 属性：在图形窗口内，在非控件区单击鼠标执行回调函数。
- WindowButtonMotionFcn 属性：鼠标在图形窗口上移动时执行的回调函数。
- WindowButtonUpFcn 属性：在图形窗口鼠标键释放时执行回调函数。
- CreateFcn 属性：初始化显示图形窗口时执行回调函数。
- DeleteFcn 属性：在删除图形窗口中执行回调函数。
- Callback 属性：用户单击图形窗口执行回调函数。
- ButtonDownFcn 属性：在图形窗口上或其周围 5 个像素范围内单击鼠标时执行回调函数。
- ResizeFcn 属性：用户调整图形窗口时执行回调函数。

2. 回调函数的编写

回调函数的编写工作首先需要进入相应控件的回调函数部分，可用鼠标右键单击控件的 View Callbacks 菜单，进入不同回调函数的程序设计部分。

下面为一按钮控件的 Callback 回调函数部分。

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

其中，第 1 行注释说明该回调函数的触发操作，第 2 行代码为回调函数的定义，函数有 3 个输入参数，对 3 个输入参数下面的 3 行做了简单的注释：参数 hObject 为控件对象的句柄；参数 eventdata 为保留参数，在以后的 MATLAB 版本中使用；参数 handles 为一个结构体，包含图形窗口中的所有对象句柄。

控件属性的获取和设置主要通过函数 get 和 set，其调用的格式如下。

- get (handles.tag1, property_name)：获取标签为 tag1 的控件的 property_name 属性。
- set (handles.tag1, property_name, property_value)：设置标签为 tag1 的控件的 property_name 属性的属性值为 property_value。

6.4.3 参数的传递

在 GUI 设计中通常需要在各个控件对象的回调函数间传递数据，传递数据的方法主要有使用数据句柄 handles、对象的 Userdata 属性和全局数据，下面具体介绍这 3 种方法。

1. 数据句柄 handles 传递参数

在前面的介绍中读者可以注意到每个回调函数的输入参数都包含 handles，参数 handles 为一结构体，用户可以把需要传递的数据保存在结构体的不同域中，实现不同回调函数间数据的传递。同时，当改变了 handles 结构体后，需要使用语句 guidata(hObject, handles) 保存、更新 handles 结构体。

【例 6.22】属性 handles 传递参数。

```
%在回调函数 pushbutton1_Callback 中写
```

```
handles.x=1:2:100;  
guidata(hObject, handles);%更新句柄  
y1=x;  
%在回调函数 pushbutton2_Callback 中写  
x= handles.x;%获取数据  
y2=2* x;
```

2. 属性 UserData 传递参数

在 GUI 设计中如果传递的参数不是很多,可以考虑直接通过对象的 UserData 属性进行各个回调函数间的数据传递工作。首先待传递的数据必须存储到控件的 UserData 属性中,在不同回调函数中需要使用这部分数据的时候直接获取控件的 UserData 属性值。

控件 UserData 属性值存储的语句如下。

set('ui_handle','UserData',Value): 其中 ui_handle 为控件的句柄,设置句柄 ui_handle 指定控件的 UserData 属性的属性值为 Value。

控件 UserData 属性值获取的语句如下。

value=get('ui_handle','UserData'): 其中 ui_handle 为获取属性值的控件的句柄,返回 ui_handle 指定控件的 UserData 属性的属性值 Value。

【例 6.23】属性 UserData 传递参数。

```
%在回调函数 pushbutton1_Callback 中写  
x=1:2:100;  
set(handles.pushbutton1, 'UserData',x) %存储数据  
y1=x;  
%在回调函数 pushbutton2_Callback 中写  
x=get(handles.pushbutton2, 'UserData')%获取数据  
y2=2*x;
```

使用这种方式传递参数虽然比较简单,但是读者可以注意到同一控件传递的数据只能有单一的变量。

3. 全局数据 global 传递参数

利用全局变量实现不同回调函数间的参数传递是最为简单的方法,只需要在需要使用该变量的地方添置变量全局定义语句 global 变量名。但是全局变量的使用不利于高效编程,特别是当程序文件比较复杂时,较多全局变量的使用,很可能引起错误。


【例 6.24】全局数据 global 传递参数。

```
%在 GUI 的 OpeningFcn 函数中写  
global x; %定义全局变量 x  
x=0:.1:2*pi;  
%在回调函数 pushbutton1_Callback 中写  
global x; %定义全局变量 x  
y1=x;  
%在回调函数 pushbutton2_Callback 中写  
global x;  
y2=2*x;
```

6.5 GUI 界面设计的实例

通过前面章节的学习相信读者对 GUI 设计的基本知识已有一定的了解,本小节将以具体的实例演示如何快速设计简单的 GUI 界面。本实例主要用于实现给定范围的正弦函数的绘制,并对绘制的曲线进行简单的编辑操作,通过本实例的学习向读者演示常用控件、菜单的设计工作。本实例中 GUI 界面布局、菜单设置通过 GUI 向导完成,而具体功能利用编程实现。

1. 创建 GUI 界面

打开 MATLAB 软件，单击主界面标题栏中的图标  (GUIDE)，如图 6.36 所示。选择新建空 GUI，并设置 GUI 保存的路径，单击“OK”按钮，进入 GUI 的设计界面。

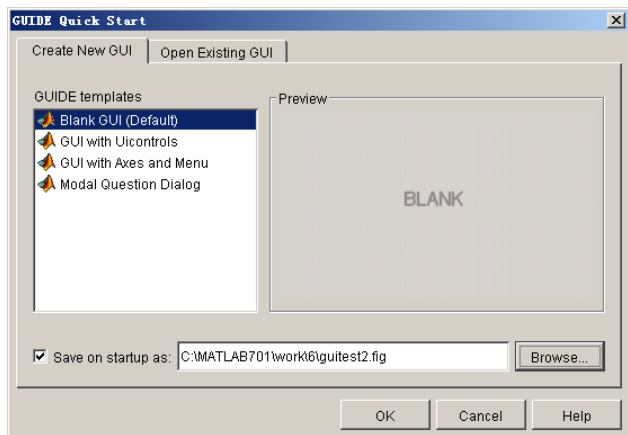


图 6.36 新建 GUI 对话框

本实例需要设计的对象如下。

- 图形窗口：作为各 GUI 对象的容器。
- 坐标轴：显示绘制的图形。
- 面板：作为设置绘制正弦函数横坐标范围的文本框的容器。
- 文本框：静态文本框提示输入的参数类型，提示信息分别为“横坐标的初值”、“横坐标的间隔”、“横坐标的终值”；与之相对应的为 3 个动态文本框，用于用户输入参数值。
- 按钮：设计 3 个按钮，分别用于执行程序、清除图形、退出程序。
- 下拉式菜单：包括“编辑”、“退出”两个菜单项，分别用于绘图的设计和程序的退出。

1) 图形窗口设计

- 通过拉伸图形窗口右下方的箭头，调整图形窗口的大小，根据设计需要调整到适合的尺寸。
- 双击图形窗口，打开图形窗口对象的属性编辑器，设置其标题属性 (name) 为“正弦函数绘制”。

2) 界面控件设计

分别拖曳坐标轴、面板、文本框、按钮到 GUI 设计区相应的位置，如图 6.37 所示。

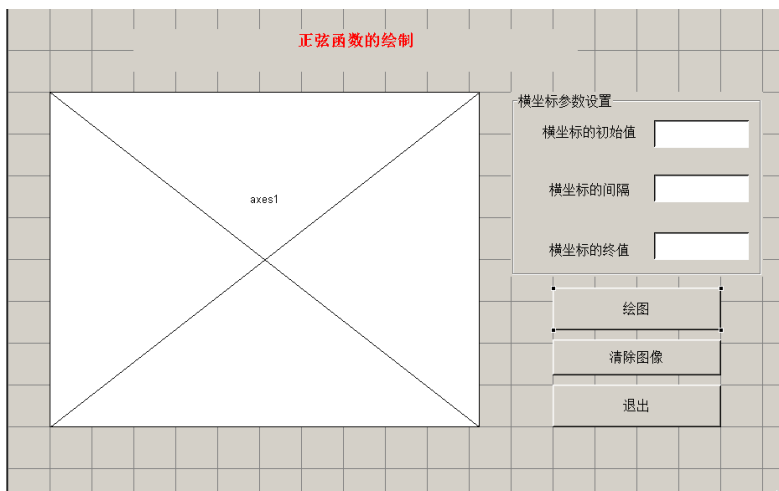



图 6.37 设计完成的 GUI 界面

分别打开各对象的属性查看器，编辑控件的属性，其中：

- 坐标轴：默认的标志 (Tag) 属性值为 axes1，可用于标识该坐标轴对象。
- 面板：设置面板显示标题属性 (Title) 的值为“横坐标参数设置”，字体大小 (FontSize) 设置为 12 号，默认的 Tag 属性值为 uipanel1。
- 静态文本框：在面板内部添加 3 个静态文本框，垂直排列，从上到下 3 个静态文本框的字符串属性 (String) 分别设置为“横坐标的初值”、“横坐标的间隔”和“横坐标的终值”，字体大小 (FontSize) 设置为 12 号，其默认使用标签 (Tag) 属性值为 text1、text2 和 text3，使用 GUI 设计的布局编辑器对齐和间隔排列 3 个静态文本框，选中 3 个静态文本框，水平方向选择左对齐和中心到中心的等间隔分布文本框；同时，在图形窗口正上方添加静态文本框，设置其字符串属性 (string) 为“正弦函数的绘制”，字体大小 (FontSize) 设置为 12 号，字体样式 (FontWeight) 属性设置为 bold，字体颜色 (ForegroundColor) 从颜色编辑器中选择红色。
- 动态文本框：在 3 个静态文本框的左面添加 3 个动态文本框，用于获取用户的输入数据，由上至下 3 个动态文本框的字符串属性 (string) 分别设置为“0”、“pi/100”和“pi”，为程序运行的默认输入参数，字体大小 (FontSize) 设置为 12 号，其默认使用标签 (Tag) 属性值为 edit1、edit2 和 edit3，同时和静态文本框的操作类似，对齐 3 个动态文本框。
- 按钮：在面板的下方添加 3 个按钮对象，分别用于控制程序的执行、清除图形和退出程序，设置其字符串属性 (string) 分别为“绘图”、“清除图像”和“退出”。

3) 菜单设计

单击 GUI 设计界面中的图标，打开菜单编辑器，在 Menu Bar 选项卡，设置下拉式菜单。其中下拉式菜单的设计如图 6.38 所示，安装菜单的层次顺序添加各菜单项，并设置各菜单项的 Lable 属性和 Tag 属性，其中 Lable 属性为界面中显示的菜单的标识，Tag 属性在程序设计中菜单唯一标识相应的菜单项。

经过上面的 GUI 设计工作后，设计好的界面如图 6.39 所示。

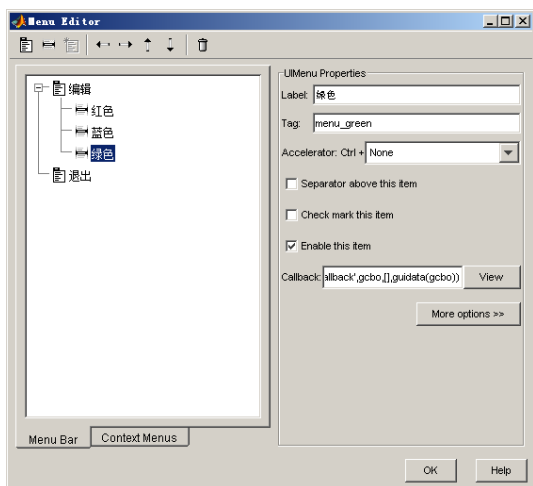


图 6.38 下拉式菜单设计

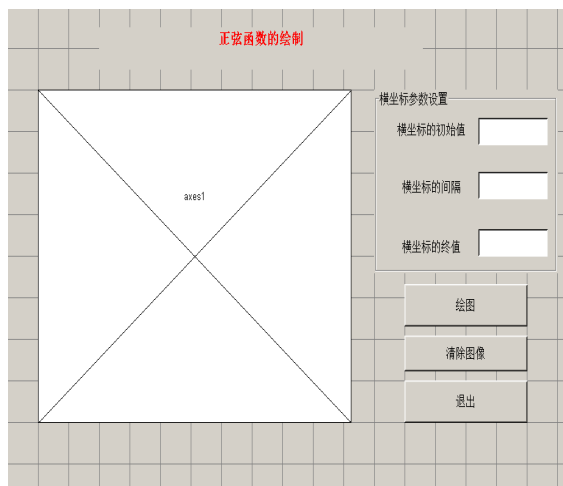


图 6.39 GUI 设计界面

2. 回调函数的编写

在完成了控件和菜单的设计工作后，进入为实现 GUI 功能的回调函数的编写工作，如图 6.40 所示。用鼠标右键单击需要编写回调函数的控件，选择“View Callbacks”→“Callback”命令，进入回调函数的编写工作中，其中部分常规的代码和注释将默认生成，用户只需按照使用的需求编写相应的代码即可。

其中，“绘图”按钮的回调函数编写如下。

```
% --- Executes on button press in pushbutton1.
function pushbutton1_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton1 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
x0=str2num(get(handles.edit1,'string'));      %从静态文本框内读入输入参数
xd=str2num(get(handles.edit2,'string'));      %从静态文本框内读入输入参数
xe=str2num(get(handles.edit3,'string'));      %从静态文本框内读入输入参数
x=[x0:xd:xe];
hlines=plot(x,sin(x));                        %绘图并返回句柄
handles.line=hlines;
guidata(hObject, handles);                    %更新保存句柄
```

“清除图像”按钮的回调函数编写如下。

```
% --- Executes on button press in pushbutton2.
function pushbutton2_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton2 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
cla;
```

“退出”按钮的回调函数编写如下。

```
% --- Executes on button press in pushbutton3.
function pushbutton3_Callback(hObject, eventdata, handles)
% hObject    handle to pushbutton3 (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
close(gcf); %关闭窗口
```

退出菜单的回调函数编写如下。

```
function menu_exit_Callback(hObject, eventdata, handles)
% hObject    handle to menu_exit (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
```

```
close(gcf); %关闭当前窗口
```

编辑→红色菜单（确认标红对否）的回调函数编写如下。


```
function menu_red_Callback(hObject, eventdata, handles)
% hObject    handle to menu_red (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.line,'color','red'); %编辑曲线的颜色为红色
```

编辑→蓝色菜单的回调函数编写如下。

```
function menu_blue_Callback(hObject, eventdata, handles)
% hObject    handle to menu_blue (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.line,'color','blue'); %编辑曲线的颜色为蓝色
```

编辑→绿色菜单的回调函数编写如下。

```
function menu_green_Callback(hObject, eventdata, handles)
% hObject    handle to menu_green (see GCBO)
% eventdata  reserved - to be defined in a future version of MATLAB
% handles    structure with handles and user data (see GUIDATA)
set(handles.line,'color','green');
```

编写完毕，单击界面上的运行图标，运行 GUI，如图 6.41 所示。其中动态文本框中显示了横坐标设置的初始值，设置完毕绘图的横坐标值，单击“绘图”按钮，将绘制一定坐标区域的正弦曲线，如图 6.42 所示。

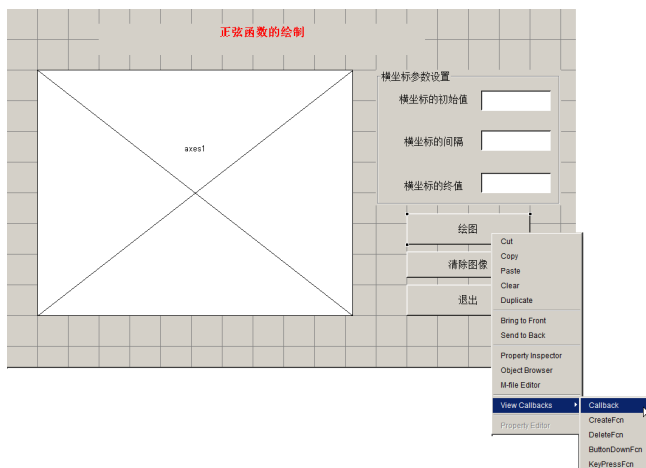


图 6.40 回调函数的编写

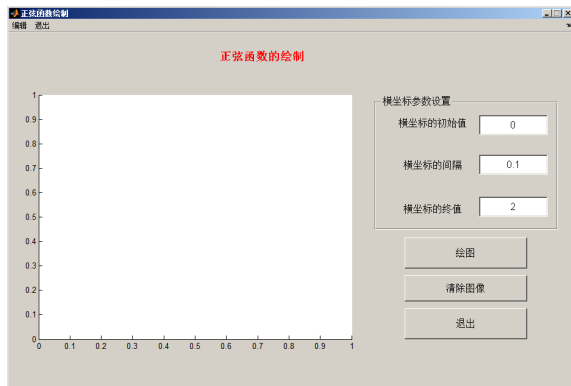


图 6.41 GUI 的运行

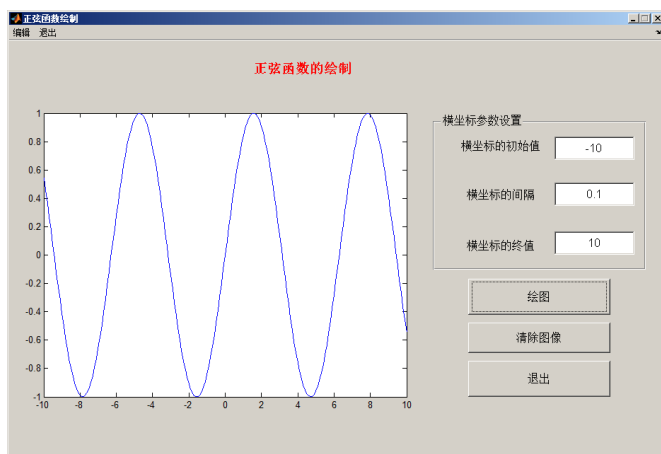


图 6.42 正弦曲线的绘制

6.6 本章小结

本章主要介绍了 MATLAB 创建 GUI 的两种方式，其中利用 GUI 向导，只需要通过简单的鼠标、键盘操作即可实现 GUI 界面的设计，设计过程中对 GUI 对象的属性设置操作也很直观，同时设计完毕会自动生成相应对象设计的 M 文件，这种方式比较适合入门读者进行一些简单的界面设计；但是对于一些比较大型的工程项目，往往涉及较多的界面设计工作，此时利用 GUI 向导创建界面，在频繁的设计调整界面的工作中，自动生成的 M 文件会存在不少冗余的代码，同时 GUI 向导的功能也没有编程方式创建 GUI 来得全面。

利用编程方式设计 GUI，编写的代码可读性强，对界面操作更为精确，易控制大型的工程，但是编程实现 GUI 并不是一件简单的事，需要掌握大量的 GUI 对象属性设置等相关函数的使用，本书对于这部分知识也仅做了简单的介绍，感兴趣的读者可以阅读相关的帮助文档。同时，用户可以注意到目前 MATLAB 主要的工具箱都提供了相应的图形界面，在学习图形界面编程的时候，我们可以阅读其 GUI 相应的 M 文件，通过实例来学习。

最后，关于用户在设计 GUI 的时候采用什么方法更好，笔者的建议是入门读者以图形界面设计（GUI 向导实现）和 GUI 功能实现（回调函数编程实现）这种方式为主，同时读者可详细阅读 MATLAB 提供的 demo 中实现 GUI 设计的代码，为今后编程实现打下基础。



第7章

数值分析

在工程实际中，往往有很多复杂的问题，我们无法通过手工计算方便地解决问题，此时借助于计算机的数值分析是处理复杂问题的有效措施。数值分析主要是用于处理科学计算的问题，而 MATLAB 软件强大的数值分析能力也是很多用户使用它的重要原因之一。MATLAB 提供了丰富的函数求解数值分析问题，特别是一些大型的复杂问题，MATLAB 数值分析功能更凸显出其优势。

本章主要介绍了数值分析技术，包括简单的数据操作、多项式运算、微分和积分、拟合和插值、线性及非线性方程组的求解。

7.1 简单的数据操作

本节将从最简单的数值分析内容讲起，主要涉及随机数的生成、描述性统计和数据排序，这部分内容虽然较为简单，但很实用，而且在很多地方都需要使用。简单的数据操作是处理很多复杂数值分析问题的基础。通过本节的学习读者将掌握数据的简单操作。

7.1.1 随机数的生成

随机数在处理实际问题中具有较好的应用。MATLAB 7.0 为用户提供了不同概率分布随机数生成的方法，其中均匀分布随机数和标准正态分布随机数生成在实际问题中较为常用。而关于其他分布随机数生成，本节也将做简单的介绍，如果读者需要使用，可以参考均匀分布随机数和标准正态分布随机数生成及相关的帮助文档。

1. 均匀分布随机数和标准正态分布随机数生成

函数 `rand()` 用于生成均匀分布在 $(0 \sim 1)$ 之间的随机数，其调用的格式如下。

- `Y = rand(n)`: 用于生成 $n \times n$ 的均匀分布随机数矩阵。
- `Y = rand(m,n)`: 用于生成 $m \times n$ 的均匀分布随机数矩阵。
- `Y = rand([m n])`: 用于生成 $m \times n$ 的均匀分布随机数矩阵。
- `Y = rand(m,n,p,...)`: 用于生成 $m \times n \times p \times \dots$ 的均匀分布随机数矩阵。
- `Y = rand([m n p...])`: 用于生成 $n \times n \times p \times \dots$ 的均匀分布随机数矩阵。
- `Y = rand(size(A))`: 用于生成与矩阵 A 相同大小的均匀分布随机数矩阵。
- `rand`: 用于生成任意的一个均匀分布随机数。

函数 `rand()` 产生的随机数为 $0 \sim 1$ 之间，同时也扩展到生成 $a \sim b$ 范围内随机数， m 个 $a \sim b$ 范围内随机数生成的表达式如下。

```
a + (b-a) * rand(1,m)
```

通过上述方法生成的指定范围内的随机数可能会有重复数据的生成，而 MATLAB 自带函数 `randperm(n)` 可用于生成 $1 \sim n$ 范围内的无重复的整数随机排列。取数列前 m 个数据，即可生成 m

个在 $1 \sim n$ 范围内无重复的随机数列。

随机数的产生需要有一个随机的种子,即利用 MATLAB 编程方式使计算机生成的随机数并不是严格意义上的随机数,只能是符合随机数分布规律的随机数。随机数生成的算法大致为通过初试的随机种子数,按照一定的算法规则,生成符合随机分布的随机数列。实际应用中由于每次生成随机数的种子数都不同,生成的随机数一般也不同,但是对于一些需要重演的案例,我们希望每次生成的随机数都能符合随机分布且固定不变,这样可以保证程序的重演性。理论上每次生成的随机数是不同的,但是利用计算机生成的随机数已不是真正的随机数,在一些教材中形象地将其称为“伪随机数”,所以我们可以通过设置生成随机数的状态,达到每次生成相同的随机数。具体的用法如下。

- `s = rand('state')`: 返回含有 35 个元素的列向量 `s`, 反映均匀分布生成器的状态, 控制随机数的生成。
- `rand('state',s)`: 设置随机数据生成的状态数为 `s`, `s` 可为任意数。
- `rand('state',0)`: 设置随机数据生成的状态数为初始状态。
- `rand('state',j)`: 设置随机数据生成的状态数为第 `j` 个状态, 其中 `j` 为整数。
- `rand('state',sum(100*clock))`: 通过 `clock` 参数控制随机数据生成的状态数, 即在每个时刻下生成的随机数不同。

【例 7.1】均匀分布随机数的生成。

```
>> y2=rand(3) %3×3 的随机矩阵生成
y2 =
    0.3412    0.3093    0.3704
    0.5341    0.8385    0.7027
    0.7271    0.5681    0.5466
>> y3=rand(2,3) %大小为 2×3 的随机矩阵生成
y3 =
    0.4449    0.6213    0.9568
    0.6946    0.7948    0.5226
>> y4=1+fix(rand(1,5)*10) %5 个 1~10 范围内随机整数的生成
y4 =
     9     2    10     3     3
>> y5=randperm(10) %1~10 范围内随机数的生成
y5 =
     4     3     6     9     7    10     8     2     1     5
>> y6=y5(1:5) %5 个 1~10 范围内随机数的生成
y6 =
     4     3     6     9     7
>> rand('state',1) %设置生成随机数的状态数为 1
>> y7=rand %生成随机数
y7 =
    0.9528
>> rand('state',1) %控制随机数的状态数仍为 1, 即每次生成相同的随机数
>> y8=rand
y8 =
    0.9528
```

函数 `randn()` 用于生成标准正态分布的随机数, 即随机矩阵的均值为 0, 方差为 1。函数的调用格式如下。

- `Y = randn(n)`: 用于生成 $n \times n$ 的标准正态分布随机数矩阵。
- `Y = randn(m,n)`: 用于生成 $m \times n$ 的标准正态分布随机数矩阵。
- `Y = randn([m n])`: 用于生成 $m \times n$ 的标准正态分布随机数矩阵。
- `Y = randn(m,n,p,...)`: 用于生成 $m \times n \times p \times \dots$ 的标准正态分布随机数矩阵。
- `Y = randn([m n p...])`: 用于生成 $n \times n \times p \times \dots$ 的标准正态分布随机数矩阵。

- $Y = \text{randn}(\text{size}(A))$: 用于生成与矩阵 A 相同大小的标准正态分布随机数矩阵。
- randn : 用于生成任意一个标准正态分布随机数。
- $s = \text{randn}('state')$: 返回含有 2 个元素的列向量 s , 反映均匀分布生成器的状态, 控制标准正态分布随机数的生成。
- $\text{randn}('state',s)$: 设置标准正态分布随机数据生成的状态数为 s , s 可为任意数。
- $\text{randn}('state',0)$: 设置标准正态分布随机数据生成的状态数为初始状态。
- $\text{randn}('state',j)$: 设置标准正态分布随机数据生成的状态数为第 j 个状态, 其中 j 为整数。
- $\text{randn}('state',\text{sum}(100*\text{clock}))$: 通过 clock 参数控制标准正态分布随机数据生成的状态数, 即在每个时刻下生成的随机数不同。

【例 7.2】标准正态分布随机数的生成。

```
>> y1=randn                                %任意标准正态分布随机数的生成
y1 =
    0.9876
>> y2=randn(3)                            %3×3 的标准正态分布随机矩阵生成
y2 =
    0.2590   -1.5290   -1.0425
   -0.3801    0.0001    1.0402
    0.1726    1.1085   -0.5620
>> y3=randn(2,3)                          %大小为 2×3 的标准正态分布随机矩阵生成
y3 =
   -1.0893   -0.6155   -0.7726
    0.2976   -1.1514    0.3583
>> y4=1+randn(1,5)*10                     %5 个 1~10 范围内标准正态分布随机数的生成
y4 =
    1.9420   -7.5191    9.7350   -3.3804   -3.2966
>> randn('state',1)                       %设置生成标准正态分布随机数的状态数为 1
>> y5=randn                                %生成标准正态分布随机数
y5 =
    0.8644
>> randn('state',1)                       %控制标准正态分布随机数的状态数仍为 1, 即每次生成相同的随机数
y6=randn
y6 =
    0.8644
```

2. 其他分布随机数生成

除了上面介绍了均匀分布随机数和标准正态分布随机数生成外, MATLAB 还提供了生成其他分布随机数的函数, 用户可以根据实际需要选择合适的随机分布, 生成随机数列。同时, 各分布函数的调用格式与函数 $\text{rand}()$ 和 $\text{randn}()$ 类似, 本书仅做简单的介绍, 感兴趣的读者可以阅读相关的帮助文档。

1) 几何分布随机数

函数 $\text{geornd}()$ 用于生成几何分布随机数, 其调用格式如下。

- $R = \text{geornd}(P)$: 生成参数为 P 的几何分布随机数。
- $R = \text{geornd}(P,m,n)$: 生成参数为 P 的几何分布随机数。

【例 7.3】几何分布随机数的生成。

```
>> g1 = geornd(0.5,1,6)
g1 =
     1     0     4     1     0     6
```

2) Beta 分布随机数

函数 $\text{betarnd}()$ 用于生成 Beta 分布随机数, 其调用格式如下。

- $R = \text{betarnd}(A,B)$: 生成参数为 A 和 B 的 Beta 随机数。
- $R = \text{betarnd}(A,B,m,n)$: 生成大小为 $m \times n$, 参数为 A 和 B 的 Beta 随机数。

【例 7.4】Beta 分布随机数的生成。

```
>> r = betarnd(2,3,1,6)
r =
    0.3202    0.6393    0.4630    0.2170    0.4779    0.1571
```

3) 正态分布随机数

函数 `normrnd()` 用于生成指定均值和方差的正态分布随机数，其调用格式如下。

`R = normrnd(MU,SIGMA)`: 生成均值为 MU，方差为 SIGMA 的正态分布随机数。

`R = normrnd(MU,SIGMA,m,n)`: 生成均值为 MU，方差为 SIGMA 的正态分布的 $m \times n$ 的随机数。

【例 7.5】正态分布随机数的生成。

```
>> n1 = normrnd(0,1)
n1 =
   -0.0376
>> n2 = normrnd(0,1,1,6)
n2 =
    0.3273    0.1746   -0.1867    0.7258   -0.5883    2.1832
n3 = normrnd(0,1,1,100);
hist(n3)                                %正态分布随机数直方图的生成
xlabel('正态分布随机数');
ylabel('频数');
```

执行上述程序，将生成如图 7.1 所示的正态分布的随机数。

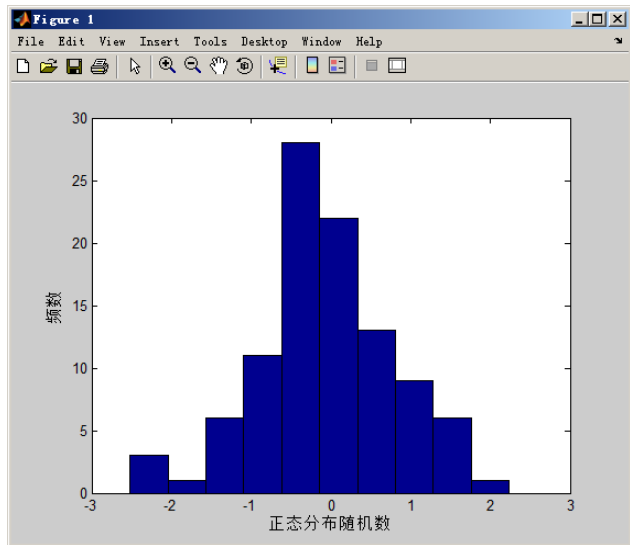


图 7.1 正态分布的随机数

4) 二项式分布随机数

函数 `binornd()` 用于生成二项式分布随机数，其调用格式如下。

- `R = binornd(N,P)`: 参数为 N 和 P 的二项式分布随机数。
- `R = binornd(N,P,m,n)`: 参数为 N 和 P 的 $m \times n$ 的二项式分布随机数。

【例 7.6】二项式分布随机数的生成。

```
>> n = 10:10:60;
r1 = binornd(n,1./n)
r1 =
     1     0     0     3     3     1
>> r2 = binornd(n,1./n,1,6)
r2 =
```

2 1 1 1 1 1 5) 卡方分布随机数

函数 `chi2rnd()` 用于生成卡方分布随机数, 其调用的格式如下。

- $R = \text{chi2rnd}(V)$: 生成自由度为 V 的卡方分布随机数。
- $R = \text{chi2rnd}(V, m, n)$: 生成自由度为 V 的 $m \times n$ 卡方分布随机数。

【例 7.7】 卡方分布随机数的生成。

```
>> r1 = chi2rnd(1:6)
r1 =
    2.4132    1.5046    3.0366    5.0328    6.5170    7.8846
>> r2 = chi2rnd(1:6,1,6)
r2 =
    0.0960    3.1481    4.6540    8.5099    2.0239    2.3271
```

6) 指数分布随机数

函数 `exprnd()` 用于生成指数分布随机数, 其调用格式如下。

- $R = \text{exprnd}(MU)$: 生成参数为 MU 的指数分布随机数。
- $R = \text{exprnd}(MU, m, n)$: 生成参数为 MU 的 $m \times n$ 指数分布随机数。

【例 7.8】 指数分布随机数的生成。

```
>> R1 = exprnd(0.5)
R1 =
    0.9870
>> R2 = exprnd(0.5,2,3)
R2 =
    0.7979    0.2523    0.8077
    0.8079    0.6506    2.0908
```

7) F 分布随机数

函数 `frnd()` 用于生成 F 分布随机数, 其调用格式如下。

- $R = \text{frnd}(V1, V2)$: 生成自由度为 $V1$ 和 $V2$ 的 F 分布随机数。
- $R = \text{frnd}(V1, V2, m, n)$: 生成自由度为 $V1$ 和 $V2$, 大小为 $m \times n$ 的 F 分布随机数。

【例 7.9】 F 分布随机数的生成。

```
>> R1 = frnd(5,10)
R2 = frnd(5,10,2,3)
R1 =
    0.5195
R2 =
    1.1295    1.6143    0.5074
    0.2637    0.5812    0.9795
```

8) Gam 分布随机数

函数 `gamrnd()` 用于生成 Gam 分布随机数, 其调用的格式如下。

- $R = \text{gamrnd}(A, B)$: 生成参数为 A 和 B 的 gam 分布随机数。
- $R = \text{gamrnd}(A, B, m, n)$: 生成参数为 A 和 B , 大小为 $m \times n$ 的 gam 分布随机数。

【例 7.10】 Gam 分布随机数的生成。

```
>> n1=gamrnd(5,10)
n2 =gamrnd(5,10,1,5)
n1 =
    77.3576
n2 =
    45.8584    54.1001    50.5419    42.7484    64.1670
```

9) 超几何分布随机数

函数 `hygernd()` 用于生成超几何分布随机数, 其调用格式如下。

- $R = \text{hygernd}(M, K, N)$: 生成参数为 M 、 K 和 N 的超几何分布随机数。

- $R = \text{hygernd}(M,K,N,m,n)$: 生成参数为 M 、 K 和 N , 大小为 $m \times n$ 的超几何分布随机数。

【例 7.11】 超几何分布随机数的生成。

```
>> h1 = hygernd(1000,40,50)
h2 = hygernd(1000,40,50,2,3)
h1 =
    4
h2 =
    0    3    1
    1    0    1
```

10) 对数正态分布随机数

函数 $\text{lognrnd}()$ 用于生成对数正态分布随机数, 其调用格式如下。

- $R = \text{lognrnd}(\text{MU}, \text{SIGMA})$: 生成参数为 MU 和 SIGMA 的对数正态分布随机数。
- $R = \text{lognrnd}(\text{MU}, \text{SIGMA}, m, n)$: 生成参数为 MU 和 SIGMA , 大小为 $m \times n$ 的对数正态分布随机数。

【例 7.12】 对数正态分布随机数的生成。

```
>> l = lognrnd(0,1,4,3)
l =
    1.0000    1.5345    1.0411
    0.7277    2.4489    1.9681
    2.9892    2.0771    1.7663
    0.1535    1.7822    0.7744
```

11) 负二项式随机数

函数 $\text{nbinrnd}()$ 用于生成负二项式随机数, 其调用格式如下。

- $\text{RND} = \text{nbinrnd}(R, P)$: 生成参数为 R 和 P 的负二项式随机数。
- $\text{RND} = \text{nbinrnd}(R, P, m, n)$: 生成参数为 R 和 P 的负二项式随机数, 大小为 $m \times n$ 的负二项式随机数。

【例 7.13】 负二项式随机数的生成。

```
>> RND = nbinrnd(3,0.01)
RND =
    339
```

12) Poisson 随机数

函数 $\text{poissrnd}()$ 用于生成 Poisson 随机数, 其调用格式如下。

- $R = \text{poissrnd}(\text{LAMBDA})$: 生成参数 LAMBDA 的 Poisson 分布随机数。
- $R = \text{poissrnd}(\text{LAMBDA}, m, n)$: 生成参数 LAMBDA 的 Poisson 分布随机数, 大小为 $m \times n$ 。

【例 7.14】 Poisson 随机数的生成。

```
>> r = poissrnd(2,1,10)
r =
    1    0    3    1    2    2    5    2    3    1
```

13) Rayleigh 随机数

函数 $\text{raylrnd}()$ 用于生成 Rayleigh 随机数, 其调用格式如下。

- $R = \text{raylrnd}(B)$: 生成参数为 B 的 Rayleigh 随机数。
- $R = \text{raylrnd}(B, m, n)$: 生成参数为 B 的 Rayleigh 随机数, 随机数的大小为 $m \times n$ 。

【例 7.15】 Rayleigh 随机数的生成。

```
>> r = raylrnd(2,2,3)
r =
    2.9841    3.2028    1.8966
    0.9181    1.6652    2.0817
```

14) t 分布随机数

函数 $\text{trnd}()$ 用于生成 t 分布随机数, 其调用格式如下。

- $R = \text{trnd}(V)$: 生成自由度为 V 的 t 分布随机数。
- $R = \text{trnd}(V,m,n)$: 生成自由度为 V 的, $m \times n$ 的 t 分布随机数。

【例 7.16】 t 分布随机数的生成。

```
>> r1= trnd(3,2,3)
r1 =
    0.2028   -1.5854    1.0123
    0.2591   -1.1181   -0.2509
```

15) 离散均匀随机数

函数 $\text{unidrnd}()$ 用于离散的均匀分布的随机数, 其调用格式如下。

- $R = \text{unidrnd}(N)$: 生成最大值为 N 的离散的均匀分布的随机数。
- $R = \text{unidrnd}(N,m,n)$: 生成最大值为 N 的离散的均匀分布的随机数, 大小为 $m \times n$ 。

【例 7.17】 离散均匀随机数的生成。

```
>> r= unidrnd(10000,2,3)
r =
    6603    2898    5341
    3420    3412    7272
```

16) 连续均匀随机数

函数 $\text{unifrnd}()$ 用于生成连续均匀随机数, 其调用格式如下。

- $R = \text{unifrnd}(A,B)$: 生成在 $A \sim B$ 范围内连续均匀随机数。
- $R = \text{unifrnd}(A,B,m,n)$: 生成在 $A \sim B$ 范围内连续均匀随机数, 随机数的大小为 $m \times n$, 包括的形式如下。

$R = \text{unifrnd}(A,B)$ $R = \text{unifrnd}(A,B,m)$ $R = \text{unifrnd}(A,B,m,n)$

例如, $\text{unifrnd}(0,1:6)$ 与 $\text{unifrnd}(0,1:6,[1 \ 6])$ 都依次生成 $[0,1]$ 到 $[0,6]$ 的 6 个均匀随机数。

【例 7.18】 连续均匀随机数的生成。

```
>> r1=unifrnd(0,1:6)
r2=unifrnd(1,10,2,3)
r1 =
    0.4449    1.3891    1.8639    3.1793    4.7842    3.1355
r2 =
    8.9213    9.8177    3.2710
    2.5566    3.4430    8.8817
```

17) Weibull 分布随机数

函数 $\text{weibrnd}()$ 用于生成 Weibull 分布随机数, 其调用格式如下。

- $R = \text{weibrnd}(A,B)$: 生成参数为 A 和 B 的 Weibull 分布随机数。
- $R = \text{weibrnd}(A,B,m,n)$: 生成参数为 A 和 B 的 Weibull 分布随机数, 大小为 $m \times n$ 。

【例 7.19】 Weibull 分布随机数的生成。

```
>> r=weibrnd(1,2,2,3)
r =
    0.6429    0.8699    0.1083
    1.1213    1.6543    0.7348
```

3. 通用函数生成各分布的随机数据

函数 $\text{random}()$ 可以生成多种分布的随机函数, 其调用格式如下。

$y = \text{random}('name',A1,A2,A3,m,n)$: 其中参数 $name$ 为指定的分布类型, $A1 \sim A3$ 为随机分布的参数, m 和 n 为随机矩阵的大小参数。参数 $name$ 的可选值包括 beta 、 binomial 、 chi-square chi2 等。

【例 7.20】 通用函数生成各分布的随机数据。

```
>> rn = random('Normal',0,1,2,3)
rn =
    0.9863    0.3274    0.0215
   -0.5186    0.2341   -1.0039
```

```
>> rp = random('Poisson',2,3)
rp =
     2     4     2
     3     2     2
     1     1     2
```

7.1.2 描述性统计参数的计算

在数据分析的过程中，首先要对数据进行描述性统计分析，以发现数据内在的规律，进而可以选择适宜的方法进行分析。常用的描述性统计参数包括数学期望（均值）、方差与偏差、最值与极差、中位数与分位数、累积和累和、协方差与相关系数、偏斜度与峰度等。本节将向读者介绍如何利用 MATLAB 的函数计算数据的描述性统计参数。

1. 数学期望与均值

数学期望又称期望或均值，是随机变量的算术平均值。在 MATLAB 中用于计算数学期望的函数为 `mean()`，其调用格式如下。

- `M = mean(A)`：计算矩阵 A 的均值，如果矩阵 A 为向量，返回向量的算术平均值，如果为矩阵，则返回矩阵 A 各列的均值。
- `M = mean(A,dim)`：生成指定维数上的矩阵 A 的均值。

【例 7.21】数学期望的计算。

```
>> A1=[2 3 4 5 7 3];           %生成向量数据
mean_A1=mean(A1)               %计算向量数据的均值
mean_A1 =
     4
>> A2=rand(3);                 %生成随机矩阵
mean_A2=mean(A2)               %默认情况下按列计算矩阵的均值
mean_A2 =
     0.4509     0.6484     0.1225
>> mean_A2=mean(A2(:))         %计算矩阵所有数的均值
mean_A2 =
     0.4073
>> A3=rand(3);
mean_A3=mean(A3,1)             %按列生成矩阵的均值
mean_A3 =
     0.5585     0.4463     0.3647
>> mean_A4=mean(A3,2)          %按行生成矩阵的均值
mean_A4 =
     0.4182
     0.2339
     0.7175
```

函数 `nanmean()` 可用于计算算术平均值，忽略其中为 NaN 的元素，其调用方法与函数 `mean()` 类似，在此不详细展开叙述，读者可以阅读相关帮助文档，并参考函数 `mean()` 的使用方法。

另外，MATLAB 还提供了一些函数计算其他类型的平均数。几何平均数是 n 个变量值连乘积的 n 次方根。而调和平均数又称倒数平均数，是变量倒数的算术平均值的倒数。

函数 `geomean()` 可计算几何平均数，调用格式如下。

- `m = geomean(X)`：生成数据 X 的几何平均数，如果 X 为向量，即返回向量的均值，如果 X 为矩阵，则返回矩阵的列均值。
- `geomean(X, dim)`：计算数据 X 指定维数上的几何平均值。

函数 `harmmean()` 可计算调和平均数，调用格式如下。

- `m = harmmean(X)`：生成数据 X 的调和平均数，如果 X 为向量，即返回向量的均值，如果 X 为矩阵，则返回矩阵的列均值。

- `harmmean(X,dim)`: 计算数据 X 指定维数上的调和平均值。

【例 7.22】各种均值的计算。

```
>> x=rand(6);
y1=geomean(x)           %矩阵 x 各列数据的几何平均值
y1 =
    0.5579    0.7297    0.4424    0.1959    0.3581    0.4417
>> y2=geomean(x,2)       %矩阵 x 各行数据的几何平均值
y2 =
    0.3554
    0.2933
    0.3551
    0.4429
    0.5416
    0.6283
>> y3=geomean(x(:))      %矩阵 x 所有数据的几何平均值
y3 =
    0.4211
>> y4=harmmean(x)        %矩阵 x 各列数据的调和平均值
y4 =
    0.5380    0.7141    0.3540    0.0580    0.2292    0.4079
>> y5=harmmean(x,1)      %矩阵 x 各行数据的几何平均值
y5 =
    0.5380    0.7141    0.3540    0.0580    0.2292    0.4079
>> y6=harmmean(x(:))     %矩阵 x 所有数据的几何平均值
y6 =
    0.1991
```

2. 方差与标准差

样本方差为样本中各数据与样本平均数的差的平方和的平均数，而样本方差的算术平方根即为样本标准差。方差与标准差用于反映数据的离散程度，方差或标准差越大，数据的离散程度越大。

方差的计算公式为：

$$\sigma^2 = \frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2$$

其中， n 为样本数， \bar{x} 为样本数据的均值。

标准差的计算公式为：

$$\text{std} = \sqrt{\frac{1}{n-1} \sum_{i=1}^n (x_i - \bar{x})^2}$$

其中， n 为样本数， \bar{x} 为样本数据的均值。

在 MATLAB 中，函数 `var()` 用于计算样本方差，其调用格式如下。

- `var(X)`: 计算数据 X 的方差，如果 X 为向量，即返回该向量的方差，而如果 X 为矩阵，则返回矩阵各列数据的方差。
- `var(X,1)`: 计算样本的方差，前置因子样本数不减 1，即为 $1/n$ 。
- `var(X,w)`: 返回以 w 为权值的数据 X 的方差。
- `var(X,w,dim)`: 返回以 w 为权值的数据 X 的方差，并指定方差计算的维度。

函数 `std()` 可用于计算样本数据的标准差，其调用格式如下。

- `s = std(X)`: 计算数据 X 的标准差，如果 X 为向量，即返回该向量的标准差，而如果 X 为矩阵，则返回矩阵各列数据的标准差。
- `s = std(X,flag)`: 计算数据 X 的标准差，flag 的值为 0 时前置因子为 $1/(n-1)$ ，其他情况下为

1/n, 默认时 flag=0。

- `s = std(X,flag,dim)`: 计算数据 X 的标准差, 参数 dim 指定标准差计算的维度, 当 dim=1 时, 计算各列数据的标准方差; 当 dim=2 时, 计算各行数据的标准方差。

函数 `nanvar()` 和 `nanstd()` 可分别用于计算数据 X 中忽略 NaN 值的方差和标准差, 其调用格式与函数 `var()` 和 `std()` 类似, 在此不再详细展开叙述。

【例 7.23】方差与标准差的计算。

```
>> x=rand(3)
x =
    0.2722    0.7468    0.4660
    0.1988    0.4451    0.4186
    0.0153    0.9318    0.8462
>> y1=var(x)                                %按列计算数据 x 的方差
y1 =
    0.0175    0.0604    0.0549
>> y2=var(x,[ ],2)                          %按行计算数据 x 的方差
y2 =
    0.0569
    0.0183
    0.2563
>> y3=var(x,1)                              %计算方差公式的前置因子为 1/n
y3 =
    0.0117    0.0402    0.0366
>> y4=var(x(:))                            %计算矩阵 x 所有数据的方差
y4 =
    0.0941
>> y5=std(x)                                %按列计算数据 x 的标准差
y5 =
    0.1323    0.2457    0.2344
>> y6=std(x,1)                             %计算标准差公式的前置因子为 1/n
y6 =
    0.1081    0.2006    0.1914
>> y7=std(x,[ ],2)                         %按行计算数据 x 的标准差
y7 =
    0.2386
    0.1352
    0.5063
>> y8=std(x(:))                            %计算矩阵 x 所有数据的标准差
y8 =
    0.3068
```

3. 最值与极差

MATLAB 提供了函数 `max()` 和 `min()` 分别用于计算数据的最大值和最小值, 两个函数的使用方法类似, 现以函数 `max()` 为例, 讲解函数的使用。函数 `max()` 的调用格式如下。

- `C = max(A)`: 计算数据 A 中的最大值, 如果 A 为向量则返回向量中的一个最大值, 如果 A 为矩阵, 则返回各列中数据最大的元素。
- `C = max(A,B)`: 比较数据 A 和 B 的大小, A 和 B 需具有相同的大小, 返回与 A 和 B 大小相同的矩阵, 返回值为各位上 A 和 B 元素较大的值。
- `C = max(A,[],dim)`: 比较指定维度上的最大值, 如果 dim 为 1 则返回各列的最值, 如果 dim 为 2 则返回各行的最值。
- `[C,I] = max(...)`: 计算数据的最值, 返回最大值 C 和最大值所在数据的位置索引。

函数 `nanmax` 和 `nanmin` 可分别用于计算忽略 “NaN” 值的数据最大值和最小值, 其调用格式与函数 `max()` 和 `min()` 类似。

函数 `range()` 可用于计算极差，即最大值和最小值的差，其调用格式如下。

- `range(a)`: 计算数据 `a` 的极差，如果 `a` 为向量，即返回该向量的最大值与最小值的差，如果 `a` 为矩阵则返回各列数据的极差。

【例 7.24】最值与极差的计算。

```
>> a=rand(3) %生成随机矩阵 a
a =
    0.6038    0.0153    0.9318
    0.2722    0.7468    0.4660
    0.1988    0.4451    0.4186

>> r1=max(a) %计算矩阵 a 各列的最大元素
r1 =
    0.6038    0.7468    0.9318

>> r2=max(a,[ ],2) %计算矩阵 a 各行的最大元素
r2 =
    0.9318
    0.7468
    0.4451

>> [r3,i3]=max(a(:)) %计算矩阵 a 的最大元素，并返回其位置
r3 =
    0.9318
i3 =
     7

>> r4=min(a) %计算矩阵 a 各列的最小元素
r4 =
    0.1988    0.0153    0.4186

>> r5=min(a,[ ],2) %计算矩阵 a 各行的最小元素
r5 =
    0.0153
    0.2722
    0.1988

>> [r6,i6]=min(a(:)) %计算矩阵 a 的最小元素，并返回其位置
r6 =
    0.0153
i6 =
     4

>> b=magic(3)/10
b =
    0.8000    0.1000    0.6000
    0.3000    0.5000    0.7000
    0.4000    0.9000    0.2000

>> max(a,b) %比较矩阵 a 和 b 的大小，返回较大值组成的新矩阵
ans =
    0.8000    0.1000    0.9318
    0.3000    0.7468    0.7000
    0.4000    0.9000    0.4186

>> r7=range(a) %计算数据 a 的极差
r7 =
    0.4050    0.7315    0.5132
```

4. 中位数与分位数

中位数是数据排序后处于中间的元素，中位数可以反映数据总体的平均状况，对于存在极端值的数据，平均数并不能很好反映数据的平均状况，而中位数可以有效避免极端值的影响。中位数的计算方法，一般先把数据排序，处于中间位置上的数即为中位数。在 MATLAB 中提供了函数 `median()` 用于计算数据的中位数，其调用格式如下。

- `M = median(A)`: 计算数据 `A` 的中位数，如果 `A` 为向量，即返回向量的中位数，如果 `A` 为矩阵，则返回矩阵各列的中位数。

- $M = \text{median}(A, \text{dim})$: 计算数据 A 指定维上的中位数, $\text{dim}=1$ 计算数据各列上的中位数, $\text{dim}=2$ 计算数据各行上的中位数。

统计学中将数据排序后, 还可以计算其分位数, 中位数即 $1/2$ 分位数, 其中四分位数在盒形图的绘制上有重要的应用。MATLAB 中计算四分位数的函数为 `quantile()`, 其调用格式如下。

- $Y = \text{quantile}(X, p)$: 函数用于计算数据 X 的 p 分位数。
- $Y = \text{quantile}(X, p, \text{dim})$: 函数用于计算数据 X 指定维上的 p 分位数, $\text{dim}=1$ 计算列分位数, $\text{dim}=2$ 计算行分位数。

【例 7.25】 中位数与分位数的计算。

```
>> x=randperm(20);
y1=median(x)                                %计算向量 x 的中位数
y1 =
    10.5000
>> y2=quantile(x,[1/4 1/2 3/4])            %计算向量 x 的 1/4、1/2、3/4 分位数
y2 =
    5.5000    10.5000    15.5000
```

5. 求积和求和

在 MATLAB 中数据求积和求和函数分别为 `prod()`和 `sum()`, 其调用格式分如下。

- $B = \text{prod}(A)$: 计算数据 A 所有元素的乘积, 如果 A 为向量, 则返回向量所有元素的乘积, 如果 A 为矩阵, 则返回各列元素的乘积。
- $B = \text{prod}(A, \text{dim})$: 指定维上的数据 A 的元素乘积, $\text{dim}=1$ 计算列元素乘积, $\text{dim}=2$ 计算行元素乘积。
- $B = \text{sum}(A)$: 计算数据 A 所有元素的和。
- $B = \text{sum}(A, \text{dim})$: 计算数据 A 指定维上所有元素的和, $\text{dim}=1$ 计算列元素的和, $\text{dim}=2$ 计算行元素的和。

【例 7.26】 数据求积和求和。

```
>> x=magic(3)
x =
     8     1     6
     3     5     7
     4     9     2
>> y1=prod(x)                                %对各列数据求积
y1 =
    96    45    84
>> y2=prod(x,2)                              %对各行数据求积
y2 =
    48
   105
    72
>> y3=sum(x)                                %对各列数据求和
y3 =
    15    15    15
>> y4=sum(x,2)                              %对各行数据求和
y4 =
    15
    15
    15
```

6. 累积和累积

在 MATLAB 中, 使用函数 `cumsum()`和 `cumprod()`可以方便地计算数据的累和和累积, 两者的调用格式类似, 现以函数 `cumsum()`讲解。函数 `cumsum()`的调用格式如下。

- $B = \text{cumsum}(A)$: 计算数据 A 的累积和, 如果 A 为向量, 即计算该向量的累积和, 如果 A

为矩阵，则计算矩阵各列元素的累和。

- $B = \text{cumsum}(A, \text{dim})$: 计算数据 A 指定维上的累积和， $\text{dim}=1$ 计算列元素的累积和， $\text{dim}=2$ 计算行元素的累积和。

【例 7.27】数据累积和累和。

```
>> x=magic(3)
x =
     8     1     6
     3     5     7
     4     9     2

>> y1=cumprod(x)           %计算数据 x 的列元素的累积
y1 =
     8     1     6
    24     5    42
    96    45    84

>> y2=cumprod(x,2)        %计算数据 x 的行元素的累积
y2 =
     8     8    48
     3    15   105
     4    36    72

>> y3=cumsum(x)           %计算数据 x 的列元素的累和
y3 =
     8     1     6
    11     6    13
    15    15    15

>> y4=cumsum(x,2)         %计算数据 x 的行元素的累和
y4 =
     8     9    15
     3     8    15
     4    13    15
```

7. 协方差与相关系数

在统计学中，协方差用于衡量两个变量的总体误差，而相关系数用于衡量两个变量间的线性关系的程度。

在 MATLAB 中计算协方差的函数为 `cov()`，其调用格式如下。

- $C = \text{cov}(X)$: 计算数据 X 的协方差，如果 X 为向量，则返回向量 X 的协方差，如果 X 为矩阵，则返回矩阵 X 的协方差矩阵，该矩阵的对角元素为数据 X 各列的方差。
- $C = \text{cov}(x,y)$: 返回数据 x 和 y 的协方差矩阵， x 和 y 需具有相同的大小。

MATLAB 提供了 `corrcoef` 函数，可以求出数据的相关系数矩阵。`corrcoef` 函数的调用格式如下。

- $R = \text{corrcoef}(X)$: 返回数据 X 的相关系数矩阵，矩阵 X 的每列数据为相关分析的一个变量，返回各列数据的相关系数。
- $R = \text{corrcoef}(x,y)$: 计算数据 x 和 y 的相关系数。

【例 7.28】通用函数生成分布的随机数据。

```
>> x=rand(3)
x =
    0.3784    0.5936    0.8216
    0.8600    0.4966    0.6449
    0.8537    0.8998    0.8180

>> y=rand(3)
y =
    0.6602    0.3412    0.3093
    0.3420    0.5341    0.8385
    0.2897    0.7271    0.5681

>> c1=cov(x)               %计算数据 x 的协方差矩阵
c1 =
    0.0763    0.0160   -0.0147
```



```

    0.0160    0.0443    0.0143
   -0.0147    0.0143    0.0102
>> c2=cov(x,y)           %计算数据 x 和 y 的协方差
c2 =
    0.0346   -0.0111
   -0.0111    0.0409
>> c3=corrcoef(x)        %计算数据 x 的相关系数矩阵
c3 =
    1.0000    0.2759   -0.5254
    0.2759    1.0000    0.6729
   -0.5254    0.6729    1.0000
>> c4=corrcoef(x,y)      %计算数据 x 和 y 的相关系数
c4 =
    1.0000   -0.2940
   -0.2940    1.0000

```

8. 偏斜度与峰度

偏斜度可用于对统计数据分布偏斜方向及程度的度量。偏斜度为 0，即可认为数据的分布是对称的，若大于 0，则数据的分布偏右，若小于 0，则数据的分布偏左。在 MATLAB 中计算数据偏斜度的函数为 `skewness()`，其调用格式如下。

- `y = skewness(X)`: 计算数据 X 的偏斜度，如果 X 为向量，则返回此向量的偏斜度，如果 X 为矩阵，则返回矩阵各列数据的偏斜度。
- `y = skewness(X,flag)`: flag 参数用于设置是否纠正偏离后再返回偏斜度，当 flag=0 时，纠正偏离，而 flag 为 1，默认状态下，不纠正偏离。

峰度用于衡量变量的集中程度，正态分布的数据峰度为 0，峰度可以衡量数据偏离正态分布的程度。函数 `kurtosis()` 用于计算数据的峰度，其调用格式如下。

- `k = kurtosis(X)`: 计算数据 X 的峰度，如果 X 为向量，则返回此向量的峰度，如果 X 为矩阵，则返回矩阵各列数据的峰度。
- `k = kurtosis(X,flag)`: flag 参数用于设置是否纠正偏离后再返回峰度，当 flag=0 时，纠正偏离，而 flag 为 1，默认状态下，不纠正偏离。
- `k = kurtosis(X,flag,dim)`: 计算指定数据维上的数据峰度。

【例 7.29】偏斜度与峰度。

```

>> x=rand(1,1000);
y1=skewness(x)           %数据 x 的偏斜度计算
y1 =
   -0.0139
>> y2=kurtosis(x)        %数据 x 的峰度计算
y2 =
    1.7779

```

7.1.3 描述性统计做图

对数据的描述性统计特征进行绘图分析可以更好地观察数据的特征，与统计值相比，绘图功能更为直观。MATLAB 为用户提供了一些观察数据描述性特征的统计图绘制功能，本节将重点讲述这些图形的绘制。

1. 盒形图

在统计学中，盒形图常用于描述数据样本的分布特征，根据数据的最大值、最小值、中位数和两个四分位数绘制而成，反映数据分布的图形。在 MATLAB 中绘制盒形图的函数为 `boxplot()`，其调用格式如下。

- `boxplot(X)`: 绘制数据 X 的盒形图，如果数据为向量，则绘制向量的盒形图，如果数据为矩

阵，则绘制矩阵各列数据的盒形图。

- `boxplot(x,G)`: 按照 `G` 变量分组，绘制盒形图。
- `boxplot(...,'Param1', val1, 'Param2', val2,...)`: 设置绘制的盒形图的参数，其中包括参数 `notch` (设置盒形图的形状，值为 1，绘制凹盒形图，值为 0，绘制矩形盒形图)、`symbol` (设置图形符号，默认为 “+”)、`orientation` (设置盒形图的方向，默认状态下为 `vertical` 垂直，`horizontal` 水平) 和 `whisker` (设置“须”图的长度，默认为 1.5)。

【例 7.30】盒形图的绘制。

```
x=rand(20,3);
boxplot(x,'orientation','horizontal');           %绘制水平方向的盒形图
figure
boxplot(x,'notch','on');                         %设置盒形图的形状为矩形
```

执行上述程序，将生成如图 7.2 和如图 7.3 所示的盒型图。

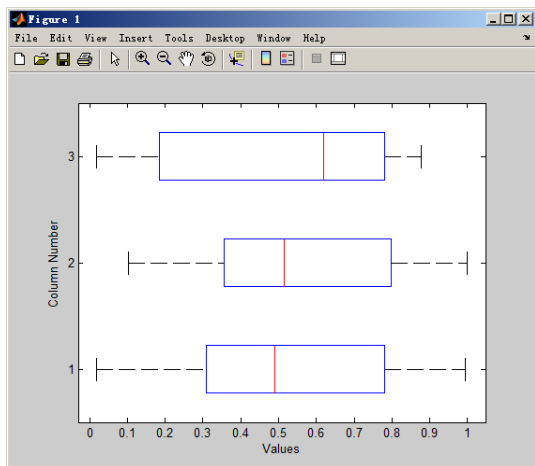


图 7.2 水平方向的盒形图

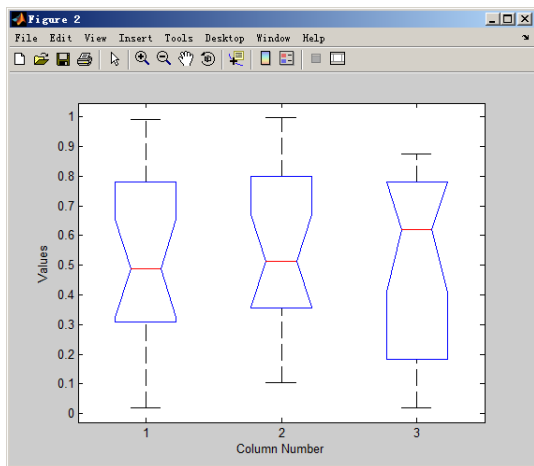


图 7.3 垂直方向的盒形图

2. 正态分布图

函数 `normplot()` 可用于绘制正态分布图，正态分布图可对数据的正态性进行检验。如果数据符合正态分布，则图形呈直线，可根据图形的线性化程度判断数据的正态性。函数 `normplot()` 的调用格式如下。

- `normplot(X)`: 绘制数据 `X` 的正态分布图，以符号 “+” 标识每个数值点的概率，叠加在数据点上的实线连接了数据的 1/4 和 3/4 分位数，点画线将此实线延伸到样本的两端，数据 `X` 为向量，则绘制该向量的正态分布图，如果为矩阵，则绘制矩阵各列数据的正态分布图。
- `h = normplot(X)`: 绘制正态分布图，并返回其句柄 `h`。

【例 7.31】正态分布图的绘制。

```
x1=normrnd(0,1,20,1);
x2=rand(20,1);
x=[x1 x2];
normplot(x);
title('正态分布图');
xlabel('数据');
xlabel('概率');
legend('正态分布数据','随机分布数据')
```

执行上述程序，将生成如图 7.4 所示的正态分布图。

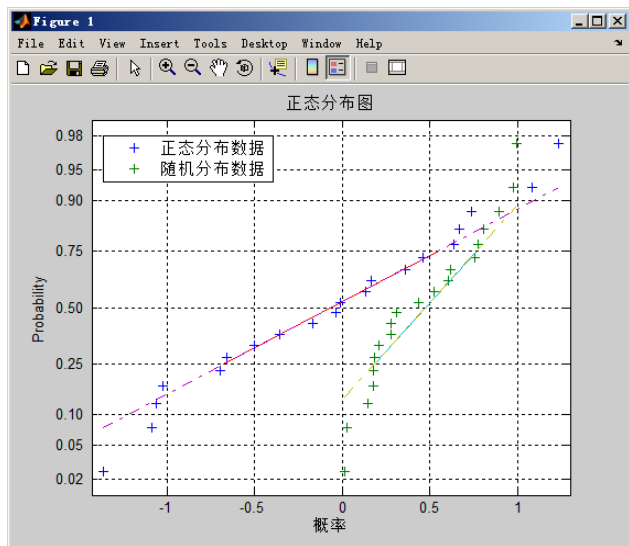


图 7.4 正态分布图的绘制

3. Q-Q 图

函数 `qqplot()` 可绘制两个样本的分位数—分位数图，用于检验数据的分布特征，如果两个样本数据来源于同一个分布则图中曲线为直线。函数 `qqplot()` 的调用格式如下。

- `qqplot(X)`: 绘制数据 `X` 的分位数—分位数图，图中样本数据点以“+”符号标识，并将位于第一分位数和第三分位数间的数据拟合成直线，并延长至样本两端。
- `qqplot(X,Y)`: 绘制数据 `X` 和 `Y` 的分位数—分位数图。
- `qqplot(X,Y,prec)`: 参数 `prec` 可用于规定分位数。

【例 7.32】 Q-Q 图的绘制。

```
x=normrnd(0,1,20,1);
y=normrnd(2,5,20,1);
qqplot(x,y);
title('分位数-分位数图');
```

执行上述程序，将生成如图 7.5 所示的分位数—分位数图。

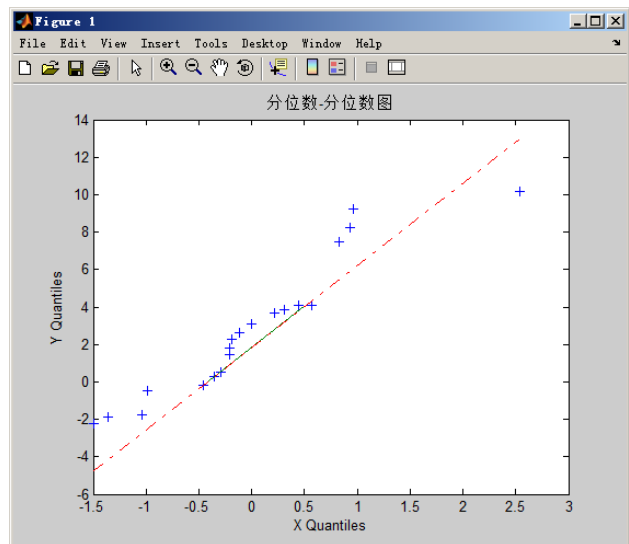


图 7.5 分位数—分位数图的绘制

4. 分组散点图

函数 `gscatter()` 可以按分类绘制分组离散点，适用于画具有多个类别的离散数据的分布图。函数 `gscatter()` 的调用格式如下。

- `gscatter(x,y,g)`: 绘制数据 x 和 y 的分组散点图， g 为分组的标记。
- `gscatter(x,y,g,'clr','sym',siz)`: 参数 `clr` 为绘图的颜色，`sym` 为各组的符号，`siz` 为符号的大小。
- `gscatter(x,y,g,'clr','sym',siz,'doleg')`: 参数 `doleg` 控制是否显示图的标记。
- `gscatter(x,y,g,'clr','sym',siz,'doleg','xnam','ynam')`: 参数 `xname` 和 `yname` 分别设置坐标轴的名称。
- `h = gscatter(...)`: 绘制分组散点图，并返回其句柄。

【例 7.33】 分组散点图。

```
load discrim
gscatter(ratings(:,1),ratings(:,2),group,'br','xo')
```

执行上述程序，将生成如图 7.6 所示的分组散点图。

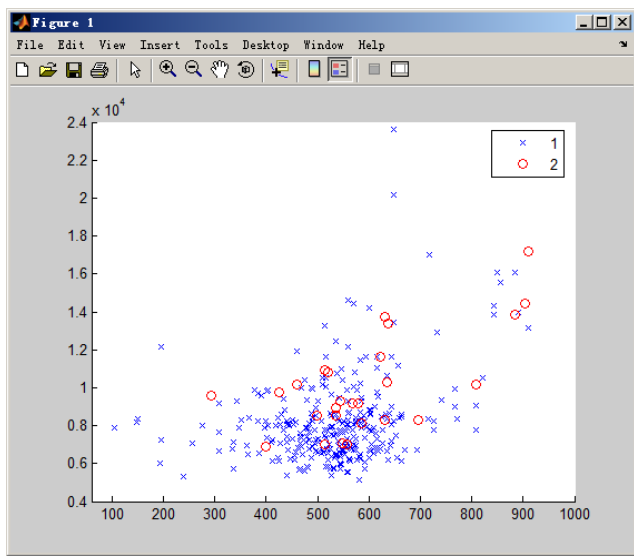


图 7.6 分组散点图的绘制

7.1.4 数据的排序

数据排序是数据分析中经常会遇到的问题，MATLAB 中提供了函数 `sort()` 和 `sortrows()` 用于数据的排序。

函数 `sort()` 的调用格式如下。

- `B = sort(A)`: 对数据 A 进行排序，如果 A 为向量，则对向量 A 进行排序，如果 A 为矩阵则对各列数据进行排序，返回排序后的数据序列。
- `B = sort(A,dim)`: 对数据 A 进行排序，并指定排序的数据维度，若 `dim=1`，则按列排，若 `dim=2`，则按行排。
- `B = sort(...,mode)`: 指定排序的模式，值为“`ascend`”按升序排序，值为“`descend`”按降序排序。
- `[B,IX] = sort(...)`: 返回排序后的数据序列和排序后的数据在原数据中的索引号。

函数 `sortrows()` 对行数据进行排序，沿列方向进行排序，其调用格式如下。

- `B = sortrows(A)`: 对数据 `A` 进行行数据排序。
- `B = sortrows(A, column)`: 依据指定 `column` 列对行数据进行排序。
- `[B, index] = sortrows(A)`: 对数据 `A` 进行行数据排序，返回排序后的结果 `B` 和对应的原数据的索引。

【例 7.34】 数据的排序。

```
>> A=magic(3)
b=sort(A)                                %对数据 A 的每一列元素进行排序
A =
     8     1     6
     3     5     7
     4     9     2
b =
     3     1     2
     4     5     6
     8     9     7
>> c=sortrows(A)                        %对数据 A 进行行排序
c =
     3     5     7
     4     9     2
     8     1     6
```

7.2 多项式运算

多项式运算是重要的数值计算内容，在一般的科学问题的求解中也往往会遇到。在 MATLAB 中，多项式由一个行向量表示，向量由按阶数降序排列的多项式系数组成，即多项式 $p(x) = a_0x^n + a_1x^{n-1} + \cdots + a_{n-1}x + a_n$ 表示为向量 $p=[a_0, a_1, \dots, a_{n-1}, a_n]$ ，在 MATLAB 中的多项式运算都是基于此向量进行的。

本节主要介绍多项式的常用运算，包括多项式求值、多项式求根、多项式乘除和多项式微积分，为后续章节内容的叙述奠定基础。

7.2.1 多项式求值

在 MATLAB 中可用函数 `polyval()` 和 `polyvalm()` 来进行多项式求值运算，可计算多项式在指定点处的值。函数 `polyval()` 和 `polyvalm()` 的区别主要在于前者是对代数多项式求值，而后者是对矩阵多项式求值。

- 函数 `polyval()` 的调用格式如下。

`y = polyval(p, x)`: 参数 `p` 为多项式的向量表示，`x` 为求值点，如果 `x` 为标量，则求多项式在该点的值，如果 `x` 为向量或矩阵，则对向量或矩阵中的每个元素进行多项式求值运算。

- 函数 `polyvalm()` 的调用格式如下。

`Y = polyvalm(p, X)`: 计算多项式 `p` 在 `X` 处的值，其中要求 `X` 为方阵。

【例 7.35】 计算多项式 $x^4 + 9x^3 + 2x^2 - 10$ 在 $x=2$ ， $x=[2 \ 3 \ 4]$ ， x 为 3×3 的单位阵时的值。

(1) 构造多项式 $x^4 + 9x^3 + 2x^2 - 10$ 的向量表达。

```
p=[1 9 2 0 -10];
```

(2) 利用函数 `polyval()` 计算 $x=2$ 时多项式的值。

```
x=2;
y=polyval(p,x)
y =
```

86
(3) 利用函数 polyval() 计算 $x=[2\ 3\ 4]$ 时多项式的值。

```
>> x=[2 3 4]
y=polyval(p,x)
x =
    2    3    4
y =
    86   332   854
```

(4) 利用函数 polyvalm() 计算 x 为 3×3 的单位阵时多项式的值。

```
>> x=ones(3);
y=polyvalm(p,x)
y =
   104   114   114
   114   104   114
   114   114   104
```

7.2.2 多项式求根

计算多项式的根, 即使多项式值为零的点, 是许多科学问题的求解中都会遇到的。在 MATLAB 中计算多项式的根极为方便, MATLAB 提供了函数 roots() 用于求取多项式的全部根, 其调用格式如下。

$x=\text{roots}(p)$: 其中 p 为多项式的系数向量, x 返回多项式 p 的所有根。

函数 roots() 可以计算出多项式的根, 而如果已知多项式的全部根, 函数 poly() 可计算出已知根的多项式, 其用法如下:

$p=\text{poly}(x)$: 其中 x 为一向量包含多项式的全部根, 通过计算返回具有根 x 的多项式的系数。

【例 7.36】 多项式求根, 计算多项式 $x^3-9x^2+26x-24$ 的根, 并验证具有该根的多项式。

(1) 计算多项式的根。

```
p=[1 -9 26 -24];
x=roots(p)
x =
    4.0000
    3.0000
    2.0000
```

(2) 计算具有该根的多项式。

```
>> p=poly(x)
p =
    1.0000   -9.0000   26.0000  -24.0000
```

7.2.3 多项式乘除

函数 conv() 用于计算多项式的乘积, 其调用格式如下。

$w = \text{conv}(u,v)$: 参数 u 和 v 为多项式的系数, 返回多项式相乘后的多项式系数 w 。

函数 deconv() 用于对多项式进行除法运算, 其调用格式如下。

$[q,r] = \text{deconv}(v,u)$: 参数 u 和 v 为多项式的系数, 返回多项式相除后的多项式系数 q 和余数 r 。

【例 7.37】 多项式乘除, 计算多项式 $3x^2+5x-2$ 与 x^2-2x+1 的乘积。

```
u=[3 5 -2];
v=[1 -2 1];
w = conv(u,v) %多项式乘法
w =
    3    -1    -9     9    -2
q= deconv(w,u) %多项式除法
q =
    1.0000   -2.0000    1.0000
```

7.2.4 多项式微积分

MATLAB 提供了函数 `polyder()` 对多项式求导，其调用格式如下。

- `k = polyder(p)`: 求多项式 p 的导函数。
- `k = polyder(a,b)`: 求多项式 $a \times b$ 的导函数。
- `[q,d] = polyder(b,a)`: 求多项式 b/a 的导函数，导函数的分子存入 q ，余数存入 d 。

函数 `polyint()` 用于对多项式求取不定积分，其调用格式如下。

- `I=polyint(p,c)`: 用于计算多项式 p 的不定积分，常数项为 c ，返回多项式系数。
- `I=polyint(p)`: 用于计算多项式 p 的不定积分，默认状态下常数项为 0。

【例 7.38】 多项式求导。

```
>> p=[2 9 -3 5];
k = polyder(p)                %求多项式 p 的导函数
k =
    6    18    -3
>> a=[2 5 3];
b=[3 4 2];
k = polyder(a,b)              %求多项式 a×b 的导函数
k =
    24    69    66    22
>> [q,d] = polyder(b,a)
q =
    7    10     2
d =
    4    20    37    30     9    %求多项式 b/a 的导函数
>> I=polyint([2,-1,0,3])
I =
    0.5000   -0.3333     0     3.0000     0
```

7.3 微分和积分

数值微分与数值积分是在数值逼近基础上发展起来的重要数值分析方法，在工程领域有很广泛的应用。

在实际问题中，往往需要根据已知的一些离散点数据，计算在某点的导数。对于此类问题的求解由于没有函数表达式，而且常规的求导方法不适用，此时可以采用数值微分的方法。数值微分是根据函数在一些离散点的函数值，推算它在某点的导数或高阶导数近似值的数值处理方法。

当我们计算某函数的定积分时，很多时候被积函数的原函数很难用初等函数表达出来；另外，在一些实际问题中，被积函数很可能是列表函数或其他形式的非连续函数，对这类函数的积分，也不能借助一般的积分方法计算原函数。数值积分是用数值逼近的方法近似计算给定的定积分值，利用 MATLAB 数值积分功能，可以快速而有效地计算复杂的积分。

本节将向读者介绍如何在 MATLAB 中处理数值微分和数值积分的问题，相信通过此处的学习读者可以熟练掌握此项重要的数值分析方法。

7.3.1 数值微分

在 MATLAB 中，并没有提供直接的计算数值导数的函数，但是可以利用函数 `diff()` 计算向前差分，实现数值微分。该方法适用于离散点比较多，数据间隔较小的场合。函数 `diff()` 的调用格式如下。

- $Y = \text{diff}(X)$: 计算离散数据 X 的向前差分, 计算公式为 $Y(i)=X(i+1)-X(i)$, $i=1,2,\dots,n-1$ 。
- $Y = \text{diff}(X,n)$: 计算离散数据 X 的 n 阶向前差分, 即在向前差分的基础上在差分, 例如 $\text{diff}(X,2)=\text{diff}(\text{diff}(X))$ 。
- $Y = \text{diff}(X,n,\text{dim})$: 在指定维度上计算离散数据 X 的 n 阶向前差分, 默认情况下 $\text{dim}=1$, 即按列计算数据 X 的向前差分, 当 $\text{dim}=2$ 时, 按行数据求导。

另外, 在 MATLAB 中除了可以通过这种差分的方法计算导数外, 还可以通过函数拟合求得离散数据的表达式, 再求导, 或者通过数据插值的方法, 进行求导。感兴趣的读者可以在阅读本书拟合与插值相关内容后, 尝试使用这些方法计算数值导数。

【例 7.40】 导数、微分的计算。

```
>> x=[5 3 5 6 8 6 8 7];
y1=diff(x)                                %一阶向前差分
y1 =
    -2     2     1     2    -2     2    -1
>> y2=diff(x,2)                            %二阶向前差分
y2 =
     4    -1     1    -4     4    -3
>> y3=diff(y1)                             %对一阶向前差分再进行一阶差分, 即二阶差分
y3 =
     4    -1     1    -4     4    -3
```

7.3.2 数值积分

计算数值积分的方法主要有简单的梯形法、辛普生法、牛顿-柯特斯法等, 其基本思想大致都是将整个积分区间分成 n 个子区间, 转化求定积分问题为分解求和问题。下面详细介绍如何利用 MATLAB 实现上述方法的数值积分。

1. 梯形法

函数 `trapz()` 基于梯形法计算以表格形式定义的函数的求定积分值, 其调用格式如下。

$z=\text{trapz}(x,y)$: 其中向量 x,y 具有函数关系 $y=f(x)$, 返回 y 在 x 各点上的积分值 z 。

【例 7.41】 利用梯形法计算数值积分。

```
>> x=1:0.001:2.5;
y=exp(-x);
z=trapz(x,y)
z =
    0.2858
```

2. 辛普生法

函数 `quad()` 用于变步长辛普生法的求定积分, 其调用格式如下。

- $q = \text{quad}(\text{fun},a,b)$: 计算函数 fun 在 $a \sim b$ 范围内的定积分值, 默认的计算精度为 $10e-6$, 返回的参数 q 为定积分值。
- $q = \text{quad}(\text{fun},a,b,\text{tol})$: 其中参数 a 和 b 为定积分的下限和上限, tol 控制积分精度。
- $q = \text{quad}(\text{fun},a,b,\text{tol},\text{trace})$: trace 控制积分过程的展现, 默认时取 $\text{trace}=0$, 表示不展现。
- $[q,\text{fcnt}] = \text{quad}(\text{fun},a,b,...)$: 返回参数 fcnt 为被积函数的调用次数。

【例 7.42】 利用辛普生法计算数值积分。

```
>> funtest=inline('exp(x)');
[q,fcnt] =quad(funtest,1,5,1e-10)
q =
    145.6949
fcnt =
    549
```


3. 牛顿—柯特斯法

函数 `quad8` 基于牛顿—柯特斯法计算数值积分，其调用格式如下。

`[l,n]=quad8('fname',a,b,tol,trace)`：其中参数的含义和函数 `quad()` 相似，读者可以参考函数 `quad()` 的解释。

【例 7.43】 利用牛顿—柯特斯法计算数值积分。

```
>> funtest=inline('exp(-x)');
[q,fcnt]=quad8(funtest,1,5,1e-10)
q =
    0.3611
fcnt =
    48
```

4. 二重积分

使用 MATLAB 函数 `dblquad()` 可以直接求出二重定积分的数值解，该函数的调用格式如下。

`I=dblquad(fun,a,b,c,d,tol,trace)`：计算函数 `fun(x,y)` 在 `[a,b]` 和 `[c,d]` 区域上的二重定积分，参数 `tol` 控制定积分计算的精度，`trace` 控制积分过程的展示。

【例 7.44】 二重积分。

```
>> f=inline('exp(-x.^2/2).*sin(x.^2+y)');
I=dblquad(f,-2,2,-1,1)
I =
    1.5745
```

7.4 拟合和插值

拟合和插值在数据处理中是非常重要的，两者都可以通过离散的函数数据点，估计出函数在任意点的值，所不同的是：拟合可以确定函数准确的表达式，通过调整函数的系数，使得函数在已知点的估计值与真实情况下已知点的值差距最小；而插值通过函数在有已知点处的取值状况，估算出函数在其他点处的近似值。

拟合和插值是离散函数逼近的有效方法，通过本节的学习读者将掌握简单的数据拟合和插值方法，今后在实际应用中可根据不同的问题采用不同的方法。

7.4.1 拟合基础

在科学实验中我们往往需要研究某些变量间是否存在什么关系，是否可以建立稳定的函数表达式，这样通过一些变量即可预测某些关键参数。于是，往往在实验中我们会获取一组离散的观测数据，如 (x_1, y_1) , (x_2, y_2) , \dots , (x_n, y_n) ，数据拟合要解决的问题即确定一函数 $y=f(x)$ 能较为准确地表达这些变量间的关系。

数据拟合主要是要找到能较好反映观测数据的近似函数，寻求整体误差最小。逼近离散数据的数值拟合常采用的拟合方法为最小二乘拟合，而根据拟合模型的类型可以分为线性拟合和非线性拟合，下面具体讲述如何在 MATLAB 中进行线性拟合和非线性拟合。

7.4.2 线性拟合

本节关于线性拟合的内容主要介绍常用的多项式拟合，其他的拟合方法将在曲线拟合章节中详细讲述。多项式拟合主要用于拟合函数表达式为多项式的数值拟合问题，即确定满足已知数据点的多项式函数的系数。在 MATLAB 中，用函数 `polyfit()` 来求解多项式的系数，之前介绍的多项

式求值函数 `polyval()` 即可方便地计算出任意点上的函数近似值。

多项式拟合函数 `polyfit()` 的调用格式如下。

- `p=polyfit(x,y,n)`: 其中 x 和 y 为待拟合数据点的值, n 为离散数据需要逼近的多项式的阶数, 返回参数 p 为拟合的多项式的系数向量, 长度为 $n+1$ 。

【例 7.45】 多项式拟合。

```
x=0:0.1:1
y=[-0.4,1.5,3.3,6,8,5.4,7.6,7.96,8.4,9.3,11.2];
plot(x,y,'k.','markersize',10)
p1=polyfit(x,y,3)
s1=polyval(p1,x)
hold on;
plot(x,s1,'-');
```

执行上述程序, 生成如图 7.7 所示的多项式拟合的效果图。

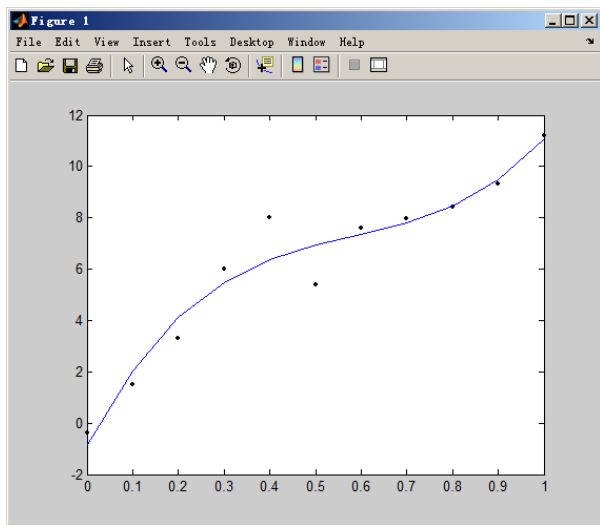


图 7.7 多项式拟合效果图

7.4.3 非线性拟合

函数 `nlfit()` 和 `lsqcurvefit()` 可用于求解 MATLAB 的非线性拟合问题, 下面详细介绍其使用方法。

函数 `nlfit()` 的调用格式如下。

- `beta = nlfit(x,y,fun,beta0)`: x 和 y 为观测数据的自变量和因变量, fun 为待拟合的模型表达式, 可以为 $y=f(x)$ 的 M 文件的函数名, 或者由 `inline()` 函数表示, β_0 是模型初始参数的估计值, 计算后获得的返回值 β 为最小二乘估计得出的模型最佳系数。
- `[beta,r,J] = nlfit(x,y,fun,beta0)`: 曲线拟合后的返回参数 r 为拟合的残差, 而 J 为残差 r 对 a 的 Jacobi 向量构成的矩阵。
- `[...] = nlfit(x, y, fun, beta0, options)`: 参数 `options` 对拟合过程进行设置, 其中包括 `MaxIter` (最大叠代次数)、`TolFun` (函数残数平方和允许值)、`TolX` (拟合系数允许的误差值) 和 `Display` (控制拟合过程的显示, 其中 `off` 表示不显示输出、`iter` 显示每次迭代的结果、`final` 只显示最终结果、`notify` 只在函数不收敛的时候显示结果)。

函数 `lsqcurvefit()` 的调用格式如下。

`[a, rnorm, r, exitflag]=lsqcurvefit(fun, a0, X,Y, lb,ub,options)`: 其中 fun 为待拟合的模型表达式, 可以为 $y=f(x)$ 的 M 文件的函数名, 或者由 `inline()` 函数表示, a_0 为模型系数的初始估计值, lb

和 ub 分别为拟合系数的预估下界和上界, 参数 options 用于拟合过程设置, 同函数 nlinfit(), 函数返回的参数中 a 为拟合估计系数, rnorm 为误差平方和, r 为拟合模型的残差, exitflag 为运行情况。

【例 7.46】非线性拟合。

(1) 创建非线性拟合函数。

```
function f=myfun1(a,x)
    f=a(1)+a(2)*exp(-0.02*a(3)*x)
```

(2) 利用函数 nlinfit() 进行非线性拟合。

```
x=1:10:100;
y=[4.54,4.99,5.35,5.65,5.90,6.10,6.26,6.39,6.50,6.59];
a0=[0.2,0.05,0.05];
options=statset('MaxIter', 100, 'TolFun', 1e-4,'Display','final')
a=nlinfit(x, y, 'myfun1', a0, options) %利用函数 nlinfit() 进行非线性拟合
f= myfun1(a,x)
```

拟合后的结果如下。

```
a =
-40.7170    45.5431   -0.0235
f =
Columns 1 through 6
    4.8476    5.0624    5.2783    5.4953    5.7132    5.9322
Columns 7 through 10
    6.1522    6.3732    6.5953    6.8184
```

(3) 利用函数 lsqcurvefit() 进行非线性拟合。

```
x=1:10:100;
y=[4.54,4.99,5.35,5.65,5.90,6.10,6.26,6.39,6.50,6.59];
a0=[0.2,0.05,0.05];
a=lsqcurvefit ('myfun1',a0,x,y) %利用函数 lsqcurvefit() 进行非线性拟合
f= myfun1(a,x)
```

拟合后的结果如下。

```
a =
-1.5718    6.4484   -0.1457
f =
Columns 1 through 6
    4.8955    5.0867    5.2835    5.4862    5.6948    5.9096
Columns 7 through 10
    6.1308    6.3585    6.5929    6.8343
```

7.4.4 插值基础

在生产实践中, 有时自变量与因变量间的函数关系并不能写出明确的函数表达式, 或者表达式很复杂, 此时为了得出函数在若干点的函数值可以通过数据插值的方法。插值是通过已知的观测点 (x_i, y_i) 建立一个简单的、连续的解析模型 $g(x)$, 用于逼近已知的观测点, 从而推测非观测点处的状况。常用的插值方法有拉格朗日插值、牛顿插值、三次样条插值、分段线性插值法等。

插值主要用于根据已知点估计未知点, 和拟合的不同点在于: 插值给出的是数据点, 而非拟合的模型; 插值函数应通过所有给定的已知数据点, 而拟合是要让数据点很好地逼近函数, 控制模型整体的误差最小。

插值问题一般分为一维插值、二维插值和高维插值问题, 下面从这几个角度详细地叙述如何利用 MATLAB 实现数据插值。

1. 一维插值

在 MATLAB 中, 函数 interp1() 可用于一维数据插值, 其调用格式如下。

- $y_i = \text{interp1}(x, y, x_i)$: 参数 x 和 y 为已知的数据点, x_i 为需要通过插值获取的点, 默认的插值

方法为 linear (线性插值), y_i 为通过一维插值函数获得的点 x_i 处的 y_i 值。

- $y_i = \text{interp1}(y, x_i)$: 插值的数据点中 x 取 $1: \text{length}(y)$, 返回插值点 x_i 处的 y_i 值。
- $y_i = \text{interp1}(x, y, x_i, \text{method})$: 指定插值的方法, 常用的插值方法有 nearest (最邻近点插值)、linear (线性插值)、spline (三次样条插值)、cubic (立方插值) 和 pchip (三次 Hermite 插值)。
- $y_i = \text{interp1}(x, y, x_i, \text{method}, 'extrap')$: 对于超过已知点 x 范围的插值点 x_i 执行外插值方法 extrap。
- $y_i = \text{interp1}(x, y, x_i, \text{method}, \text{extrapval})$: 对于超过已知点 x 范围的插值点 x_i 取确定的插值点, 一般为 NaN 或 0。

下面对各种数据插值方法做简单介绍。

- nearest (最邻近点插值): 插值点估计函数值为已知点中最接近插值点的值, 该算法是最快的插值方法, 但是数据平滑最差, 插值后的数据是不连续的。
- linear (线性插值): 插值点估计函数值通过线性函数计算所得, 该算法执行速度较快, 有较高的插值精度, 为插值函数的默认设置。
- spline (三次样条插值): 插值点估计函数值通过三次多项式计算所得, 该算法精度高, 能获得最平滑的插值曲线, 但执行速度最慢。
- cubic (立方插值) 或 pchip (三次 Hermite 插值): 插值点估计函数值通过 Hermite() 函数计算所得, 该算法较慢, 但精度高, 平滑度好。

综上, 不同的插值算法的原理不同, 可能会得到不同精度的解, 在使用中用户应根据问题的实际需要选择合适的插值方法。

【例 7.47】一维插值。

```
x=1:10:100;  
y=[4.54,4.99,5.35,5.65,5.90,6.10,6.26,6.39,6.50,6.59];  
t=1:1:100;  
y1=interp1(x,y,t,'nearest');  
subplot(2,2,1);  
plot(x,y,'r*',t,y1)  
title('最邻近点插值')  
y2=interp1(x,y,t,'linear');  
subplot(2,2,2);  
plot(x,y,'r*',t,y2,'b:')  
title('线性插值')  
y3=interp1(x,y,t,'spline');  
subplot(2,2,3);  
plot(x,y,'r*',t,y3)  
title('样条插值')  
y4=interp1(x,y,t,'cubic');  
subplot(2,2,4);  
plot(x,y,'r*',t,y4,'k-')  
title('三次插值')
```

执行上述程序, 生成如图 7.8 所示的不同的一维插值效果图。

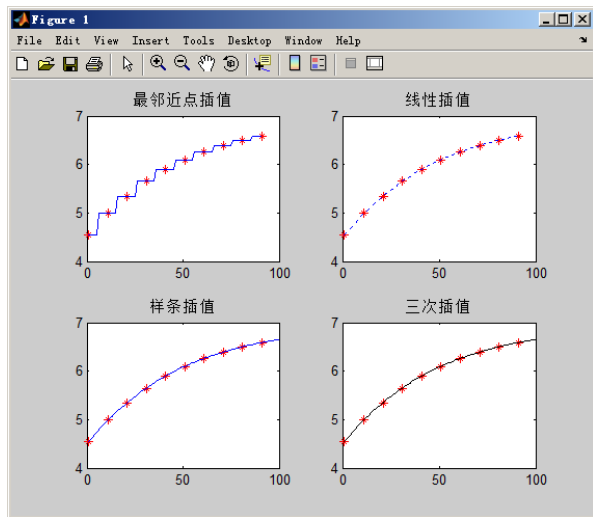


图 7.8 不同的一维插值效果

2. 二维插值

MATLAB 中提供了函数 `interp2()` 和 `griddata()` 用于二维数据的插值，其中函数 `interp2()` 主要用于二维网格数据的插值，而函数 `griddata()` 可对任意分布的数据进行二维插值。

函数 `interp2()` 的调用格式如下。

- `zi = interp2(x,y,z,xi,yi)`: x 和 y 为已知数据点在 x 轴和 y 轴上的坐标点， z 为相应已知点上的函数值，要求已知点 x 和 y 大小相同、单调， xi 和 yi 为插值点，要求 xi 和 yi 必须在已知点 x 和 y 范围内，如果超出范围，将得到“NaN”值，参数 zi 为插值函数范围的插值结果。
- `zi = interp2(z,xi,yi)`: x 和 y 默认情况下分别为数据 z 对应的行号和列号。
- `zi = interp2(x,y,z,xi,yi,method)`: 参数 `method` 用于指定二维插值的方法，包括 `nearest` (最近邻点插值)、`linear` (线性插值，默认)、`spline` (三次样条插值) 和 `cubic` (立方插值)。
- `zi = interp2(...,method,extrapval)`: 设置当插值点超过已知点后的插值的值为 `extrapval`。

【例 7.48】 二维插值网格插值。

```
[x,y,z]=peaks(10);
mesh(x,y,z)
[xi,yi]=meshgrid(-10:0.5:10,-10:0.5:10);
z1=interp2(x,y,z,xi,yi,'nearest');
z2=interp2(x,y,z,xi,yi,'linear');
z3=interp2(x,y,z,xi,yi,'spline');
z4=interp2(x,y,z,xi,yi,'cubic');
subplot(2,2,1)
mesh(xi,yi,z1)
title('最近点插值')
subplot(2,2,2)
mesh(xi,yi,z2)
title('线性插值')
subplot(2,2,3)
mesh(xi,yi,z3)
title('样条插值')
subplot(2,2,4)
mesh(xi,yi,z4)
title('三次插值')
```

执行上述程序，生成如图 7.9 所示的不同的二维网格插值效果图。

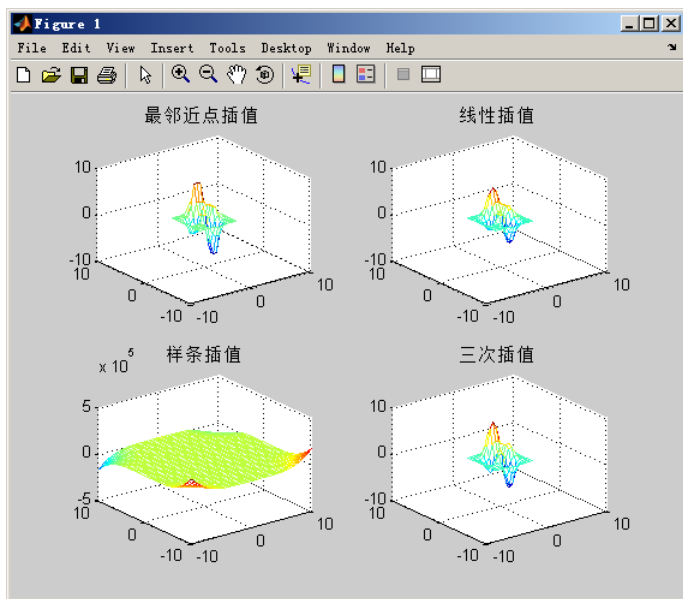


图 7.9 不同的二维网格插值效果

函数 `griddata()` 的调用格式如下。

- `zi = griddata(x,y,z,xi,yi)`: `x` 和 `y` 是已知数据组, `z` 是相对应的已知点上的函数值, `xi` 和 `yi` 是需要插值的数据点, `zi` 为函数返回的插值结果。
- `[...] = griddata(...,method)`: 设置二维数据插值的方法包括 `nearest` (最近点插值)、`linear` (线性插值, 默认)、`cubic` (立方插值) 和 `v4` (MATLAB 4.0 版本中提供)。

【例 7.49】 利用函数 `griddata()` 进行二维插值。

```
[x,y]=meshgrid(-10:2:10,-15:1:10);
z=x.^2+y.^2;
[xi,yi]=meshgrid(-2:0.1:2,-5:0.1:5);
z1=griddata(x,y,z,xi,yi,'nearest');
z2=griddata(x,y,z,xi,yi,'linear');
z3=griddata(x,y,z,xi,yi,'cubic');
z4=griddata(x,y,z,xi,yi,'v4');
subplot(2,2,1)
surf(xi,yi,z1)
title('最近点插值')
subplot(2,2,2)
surf(xi,yi,z2)
title('线性插值')
subplot(2,2,3)
surf(xi,yi,z3)
title('三次插值')
subplot(2,2,4)
surf(xi,yi,z4)
title('v4 插值')
```

执行上述程序, 生成如图 7.10 所示的不同的二维插值效果图。

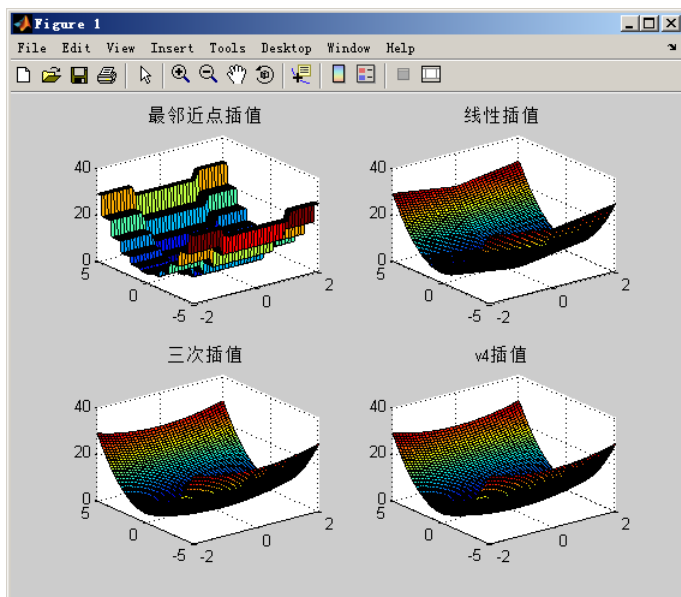


图 7.10 不同的二维插值效果

3. 高维插值

MATLAB 支持三维插值及三维以上的插值，可以通过函数 `interp3`（三维插值函数）、`interp`（ n 维插值函数）和 `ndgrid`（ n 维网格数据插值）实现。其使用方法与 `interp2` 类似，直接通过实例向读者演示这些函数的使用。

【例 7.50】高维插值。

```
[x,y,z,v] = flow(10);
[xi,yi,zi] = meshgrid(-1:0.05:1, -1:0.05:1, -1:0.05:1);
vi = interp3(x,y,z,v,xi,yi,zi);
slice(xi,yi,zi,vi,[0.6 0.8],0.5,[-0.2 0.2])
```

执行上述程序，生成如图 7.11 所示的高维插值效果图。

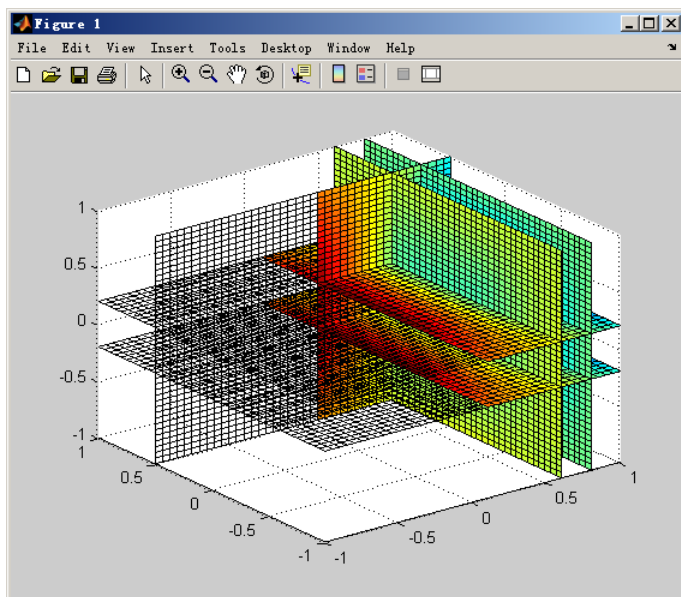


图 7.11 高维插值效果

7.5 线性方程组的求解

MATLAB 是一个功能强大的线性方程组求解工具，它特别适合求解大规模的线性方程组，MATLAB 中求解线性方程组的方法主要有如下几种。

1. 利用左除运算符“\”求解

对于线性方程组 $Ax=b$ ，可以通过运算符“\”求解，即 $x=A \backslash b$ 。

2. 矩阵求逆

对于线性方程组 $Ax=b$ ，如果矩阵 A 为可逆的方阵，则可以通过矩阵求逆的方法求解线性方程组，即 $x=\text{inv}(A)*b$ 。

3. 利用函数 `linsolve()` 求解

`linsolve()` 的调用格式如下。

$X = \text{linsolve}(A,b)$ ：其中 A 为方程组的系数， b 为方程的常数项，返回的 X 为方程组的解。

【例 7.51】 线性方程组的求解。

求解线性方程组：

$$\begin{cases} x_1 + 2x_2 + 3x_3 = 14 \\ x_1 - 2x_2 + x_3 = 0 \\ 2x_1 + 3x_2 - 3x_3 = 1 \end{cases}$$

```
>> A=[1 2 3; 1 -2 1; 2 3 -3];
b=[14;0;1];
X1=A\b                                %利用左除运算符“\”求解
X1 =
    1.4706
    2.1176
    2.7647
>> X2=inv(A)*b                        %矩阵求逆
X2 =
    1.4706
    2.1176
    2.7647
>> X3 = linsolve(A,b)                %利用函数 linsolve() 求解
X3 =
    1.4706
    2.1176
    2.7647
```

7.6 非线性方程求解

在科学计算中，经常遇到求解非线性方程的问题。函数 `fzero()` 可用于非线性方程的求解，而函数 `fsolve()` 可用于非线性方程组的求解。在 MATLAB 中求解的非线性方程，需要化为标准形式 $f(x)=0$ 。

函数 `fzero()` 的调用格式如下。

- $x = \text{fzero}(\text{fun}, x_0)$ ：fun 为求解的线性方程的函数表达式， x_0 为方程解的初始估计值， x 为计算所得的方程的解。
- $x = \text{fzero}(\text{fun}, x_0, \text{options})$ ：参数 options 用于设置求解方程组的参数，通过 `options=optimset`

('fzero') 可以获取默认值。

- `[x,fval] = fzero(...)`: 返回的参数 `fval` 为函数 `fun` 在 `x` 点处的值。
- `[x,fval,exitflag] = fzero(...)`: 返回参数 `exitflag` 反映求解线性方程的情况, 若 `exitflag` 大于 0 则表示找到函数零点, 若 `exitflag` 小于 0 则表示未找到函数的零点。

【例 7.52】 非线性方程的求解。

求解线性方程 $x^3 - 2x^2 + e^x - 5 = 0$:

```
>> x=fzero('x^3-2*x^2+exp(x)-5',3)
x =
    1.7514
```

函数 `fsolve()` 的调用格式如下。

- `x = fsolve(fun,x0)`: 计算方程组 `fun` 的解, 其中 `x0` 为解的初始估计值, `x` 为返回的方程组的解。
- `x = fsolve(fun,x0,options)`: 参数 `options` 对线性方程组解的求解进行相关设置。
- `[x,fval] = fsolve(...)`: 返回参数 `fval` 为解 `x` 对应的函数值。
- `[x,fval,exitflag] = fsolve(...)`: 返回参数 `exitflag` 反映求解线性方程组的情况, 若 `exitflag` 大于 0 则表示找到函数零点, 若 `exitflag` 小于 0 则表示未找到函数的零点。
- `[x,fval,exitflag,output] = fsolve(...)`: 参数 `output` 为求解非线性方程组的相关信息。

【例 7.53】 非线性方程组的求解。

求解非线性方程组

$$\begin{cases} x_1 + 2e^{x_2} = 4 \\ x_1 - 2e^{x_2} = 0 \end{cases}$$

(1) 创建该线性方程组的函数表达。

```
function y=fc(x)
y(1)=x(1)+2exp(x(2))-4;
y(2)=x(1)-2exp(x(2));
y=[y(1),y(2)];
```

(2) 解线性方程组。

```
>> [x,fval,exitflag,output]=fsolve('fc', [1,1]) %利用函数 fsolve() 求解线性方程组
Optimization terminated: first-order optimality is less than options.TolFun.
x =
    2.0000    0.0000
fval =
    1.0e-011 *
    0.2401   -0.2401
exitflag =
     1
output =
    iterations: 5
    funcCount: 18
    algorithm: 'trust-region dogleg'
    firstorderopt: 9.6043e-012
    message: [1x76 char]
```

7.7 本章小结

本章主要介绍了 MATLAB 的数值分析功能，首先介绍了最为基础的简单数据操作部分，包括随机数的生产、数据的描述性特征获取、数据排序等相关知识，这部分的内容是后续章节学习的重要基础；然后，向读者介绍了常用的数值分析方法，包括多项式运算、数值微分和积分、拟合和插值、线性和非线性方程组的求解。其中，多项式运算在很多算法中都会涉及，而数值微分和积分、拟合和插值更是工程应用领域解决各种问题时普遍会用到的，学习如何在 MATLAB 中对线性和非线性方程组的求解，可以帮助我们快速而有效地解决很多复杂问题。

通过本章的学习将使读者了解、掌握各种数值分析的方法，今后在遇到同类问题时可以快速地选择合适的算法，通过 MATLAB 实现。



第8章

符号计算功能

MATLAB 的科学运算主要分为数值运算与符号运算两类，在第 7 章中已介绍了数值运算的相关内容，本章将重点讲述符号运算的内容。科学领域中的很多问题都需要通过各种公式推导，此时需要借助符号计算来解决问题。符号计算是对字符符号对象进行科学的运算与分析，这是与传统的数值运算的最大区别。

MATLAB 专门提供了符号计算工具箱用于求解符号计算问题，可见 MATLAB 具有强大的符号计算功能。同时，在 MATLAB 中使用符号计算功能操作简单，在用户群体中广为应用。

8.1 符号计算概述

符号计算是指以符号形式进行的计算，主要用于研究科学问题的算法求解。符号计算与数值计算最大的差异是符号计算的对象为符号对象，因而在实现符号计算前用户需要首先定义符号对象。MATLAB 中的符号计算主要通过符号工具箱（Symbolic Math Toolbox）来实现。

在 MATLAB 中提供的符号计算功能如下。

- 符号表达式的化简、合并与分解、代数运算、分子分母提取、自变量的确定。
- 符号矩阵的代数运算、特殊运算。
- 符号极限、符号微分、符号积分和符号级数。
- 符号方程求解：代数方程、微分方程。
- 符号函数图形绘制：二维图形、三维图形。
- 符号对象与数值对象的转换。

8.2 符号对象的创建

在 MATLAB 中的符号计算主要是对符号对象进行操作的，符号对象为用来存储字符符号的数对象，在使用符号计算功能前，首先需要创建符号对象。本节主要介绍符号对象的创建，其中常用的符号对象主要包括符号常量和变量、符号表达式、符号矩阵。

8.2.1 符号变量

在 MATLAB 中创建符号常量和变量的函数为 `sym()` 和 `syms()`，两函数的主要区别在于前者每次只能创建一个符号变量，而后者可以同时创建多个符号变量。

函数 `sym()` 的调用格式如下。

- `x = sym('x')`: 创建符号变量 `x`，无须对变量 `x` 赋值，在以后的运算中直接对符号变量 `x` 进

行操作, 返回的结果为带符号的表达式, 而非数值结果。

- `x = sym('x','real')`: 创建符号变量 `x`, 并设置其为实体型。
- `x = sym('x','unreal')`: 创建符号变量 `x`, 并设置其为非实体型。

【例 8.1】符号变量的创建。

```
>> x=sym('x'); %定义符号变量 x 和 y
y=sym('y');
>> class(x) %检查变量 x 的数据类型为符号变量
ans =
sym
>> z=x*x+y*y %符号变量的简单运算, 对符号对象操作, 返回符号表达式
z =
x^2+y^2
%数值运算的比较
>> a=5;
b=5;
c=a*a+b*b %注意与符号计算的区别, 数值运算返回的是计算的数值结果
c =
50
```

函数 `sym()` 每次只能创建一个符号变量, 为了便于用户一次定义多个符号变量, MTALAB 提供了 `syms()` 函数, 其调用格式如下。

- `syms arg1 arg2 ...`: 创建符号变量 `arg1`、`arg2`。
- `syms arg1 arg2 ... real`: 创建实体型的符号变量 `arg1`、`arg2`。
- `syms arg1 arg2 ... unreal`: 创建非实体型的符号变量 `arg1`、`arg2`。

【例 8.2】多个符号变量的创建。

```
>> syms x y
z=x*x+y*y
z =
x^2+y^2
```

8.2.2 符号常量

符号常量是不含变量的符号对象, 符号常量看上去类似于数值常量, 但与一般常量不同的是, 对符号常量的操作, 可以得到更为精确的符号解, 而非数值运算的近似解。

符号常量的定义也使用函数 `sym()`, 当函数 `sym()` 的输入对象为常数或常数的字符串时, 即构建了一个符号常量, 同时对于符号常量也可以进行数值精度的设置, 控制符号常量返回的精度。

`S = sym(A,flag)`: 其中参数 `A` 为创建的符号常量, 参数 `flag` 用于设置符号常量的返回精度, 参数值可取 `d` (最接近的十进制浮点数)、`f` (最接近的十六进制浮点数)、`r` (最接近的有理数形式, 默认) 和 `e` (最接近的机器浮点误差的有理值)。

【例 8.3】符号常量的创建。

```
>> x=sym(1/5) %创建符号常量
x =
1/5
>> y=sym(1/4)
y =
1/4
>> z=x+y %符号常量的计算结果为精确的值
z =
9/20
>> m=1/5+1/4 %数值运算的结果为带小数的值
m =
0.4500
>> a=sym(2/3,'r') %返回符号常量最接近的有理数形式
a =
```

[illegible]

8.2.3 符号表达式

符号表达式为含有符号对象（符号常量、符号变量）的表达式，其创建方法如下。

1. 利用函数 `sym()` 直接创建

函数 `sym()` 可用于直接创建符号表达式，当函数的输入参数为表达式，即可创建符号表达式，其调用格式如下。

sym(A): 其中 A 为字符串的表达式，必须被单引号引用。

2. 利用符号对象创建

符号表达式也可以通过创建的符号对象来实现，当把已定义的符号变量或者符号常量连接为表达式，即可完成符号表达式的创建。

【例 8.4】符号表达式的创建。

```
>> y1=sym('cos(x)') %利用函数 sym() 直接创建
y1 =
cos(x)
>> x=sym('x')
x =
x
>> y2=cos(x) %利用已创建的符号变量 x 创建符号表达式
y2 =
cos(x)
```

8.2.4 符号矩阵

由符号对象构建的矩阵为符号矩阵，符号矩阵的格式与一般的数据矩阵类似，其创建方法如下。

1. 利用函数 `sym()` 直接创建

函数 `sym()` 的输入为符号矩阵，矩阵各元素可以为符号常量、符号变量或者符号表达式，各元素的长度不要求一样长。

2. 利用已有的符号对象组合创建

与一般数据矩阵的定义类似，通过直接创建的方法，矩阵行间元素以空格或逗号相隔，列间以分号相隔，矩阵的元素为已定义的符号对象。

【例 8.5】 符号矩阵的创建。

```
>> A=sym('[x 5/x 6;sin(x) y x+y]') %利用函数 sym() 直接创建
A =
[ x, 5/x, 6]
[ sin(x), y, x+y]
>> x=sym('sin(x)');
y=sym('1/5');
B=[1 x;y 2] %利用已有的符号对象组合创建
B =
```

```
[ 1, sin(x)]  
[ 1/5, 2]
```

8.2.5 符号函数

在 MATLAB 中创建符号函数的方法主要有如下两种。

- 符号表达式法：创建符号表达式，赋值的变量即为符号函数。
- M 文件的方法：将需要创建符号函数的函数编辑为独立的 M 文件，在使用的时候直接调用该 M 文件即可。

对于比较复杂的符号函数，建议使用 M 文件的方式创建，而符号表达式的方法适合用于需要快速创建简单符号函数的场合。下面以具体的实例详细叙述符号函数的使用。

【例 8.6】符号函数的创建。

符号表达式法如下。

```
>> syms x y  
f=x^2+y^2  
f =  
x^2+y^2
```

M 文件的方法如下。

(1) 首先，编写需要创建为符号函数的 M 文件。

```
function y=f1(x)  
if x<0  
    y=abs(x);  
else  
    y=cos(x)+x;  
end  
end
```

(2) 定义该函数为符号函数。

```
>> f1=sym('f1(x)')  
f1 =  
f1(x)
```

(3) 使用该符号函数。

```
>> subs(f1,-2)  
ans =  
2  
>> subs(f1,5)  
ans =  
5.2837
```

8.3 符号表达式的基本操作

符号表达式为重要的符号对象，符号运算的很多操作都是基于符号表达式进行的。本节将主要介绍符号表达式的基本操作，包括符号表达式的化简、符号表达式的合并与分解、符号表达式的代数运算、符号表达式的分子分母提取、符号表达式的自变量的确定。

8.3.1 符号表达式的化简

对于一些比较复杂的符号表达式，MATLAB 提供了函数 `simplify()` 用于符号表达式化简，可以方便用户阅读符号表达式。函数 `simplify()` 的调用格式如下。

`B = simplify(A)`: 输入参数 `A` 为需要化简的符号表达式，返回参数 `B` 为化简后的表达式。

函数 `simple()` 也可以实现符号表达式的化简，且最后化简的表达式具有最短的函数表达。函数

simple()的调用格式如下。

- $r = \text{simple}(S)$: 输入参数 S 为待化简的函数表达式, 返回化简后的表达式 r 。
- $[r, \text{how}] = \text{simple}(S)$: 返回化简后的表达式 r 和表达式化简的方法 how , 其中提供的化简方法有 `simplify` (对表达式进行化简)、`radsimp` (对含根式的表达式化简)、`combine` (对表达式中以和、积、幂等形式出现的项化简)、`collect` (对表达式中同类项合并)、`factor` (对表达式因式分解) 和 `convert` (表达式形式转换)。

另外, 对于含有相同字符串内容的复杂符号表达式, 可以利用函数 `subexpr()` 把符号表达式中相同的字符串简化为一个字符来代替。函数 `subexpr()` 的调用格式如下。

- $[Y, \text{SIGMA}] = \text{subexpr}(X, \text{SIGMA})$: 以字符变量 SIGMA 替换符号表达式 X 中重复的字符串, 返回替换后的字符表达式 Y , SIGMA 为替换的重复字符串。
- $[Y, \text{SIGMA}] = \text{subexpr}(X, 'SIGMA')$: 该调用格式与上面的不同之处在于参数 SIGMA 上加了单引号, 其他功能类似。

【例 8.7】符号表达式的化简。

(1) 利用函数 `simplify()` 实现符号表达式的化简。

```
>> s=sym('x^3+2*(x-y)+y*(x+3*x)'); %创建符号表达式
y= simplify(s) %化简符号表达式
y =
x^3+2*x-2*y+y*(4*x)
>> s=sym('x^3+3*x^2+3*x+1');
y = simplify(s)
y =
x^3+3*x^2+3*x+1
>> s=sym('(x+1)*x*(x-1)');
y = simplify(s)
y =
x*(x^2-1)
```

(2) 利用函数 `simple()` 实现符号表达式的化简。

```
>> s=sym('x^3+2*(x-y)+y*(x+3*x)');
[r,how] = simple(s) %以 simplify 方式进行符号表达式化简
r =
x^3+2*x-2*y+y*(4*x)
how =
simplify
>> s=sym('x^3+3*x^2+3*x+1');
[r,how] = simple(s) %以 factor 方式进行符号表达式化简
r =
(x+1)^3
how =
factor
>> s=sym('(x+1)*x*(x-1)');
[r,how] = simple(s) %以 combine 方式进行符号表达式化简
r =
x^3-x
how =
combine(trig)
```

(3) 利用函数 `subexpr()` 实现符号表达式的化简。

```
>> t = solve('a*x^3+b*x^2+c*x+d = 0');
[r,s] = subexpr(t,s)
r =

1/6/a*s^(1/3)-2/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a

-1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a+1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2/3*(3*c*a-b^2)/a/s^(1/3))
```

```
-1/12/a*s^(1/3)+1/3*(3*c*a-b^2)/a/s^(1/3)-1/3*b/a-1/2*i*3^(1/2)*(1/6/a*s^(1/3)+2/3*(
3*c*a-b^2)/a/s^(1/3))
s =
36*c*b*a-108*d*a^2-8*b^3+12*3^(1/2)*(4*c^3*a-c^2*b^2-18*c*b*a*d+27*d^2*a^2+4*d*b^3)^(
1/2)*a
```

8.3.2 符号表达式的合并与分解

MATLAB 中提供了一系列的函数可对符号表达式进行合并和分解, 用户可根据实际需要选择相应的函数操作, 下面简要介绍这些函数的使用方法。

1. 合并同类项

函数 `collect()` 用于合并符号表达式中的同类项, 其调用格式如下。

- `R = collect(S)`: 根据符号表达式 `S` 默认的自变量进行同类项合并。
- `R = collect(S,v)`: 合并字符变量 `v` 的同类型。

【例 8.8】 合并同类项。

```
>> S=sym('x*cos(y)+y*x+y');
collect(S) %合并字符变量 x 的同类型
ans =
(1+y)*x+cos(y)+y
>> collect(S,'y') %合并字符变量 y 的同类型
ans =
(x+1)*y+x+cos(y)
```

2. 展开

函数 `expand()` 可用于对符号表达式进行展开, 其调用格式如下。

`R = expand(S)`: 展开符号表达式 `S` 为 `R`。

【例 8.9】 符号表达式的展开。

```
>> S=sym('x*(2*x-y)+y*(x+y)'); %创建符号表达式
expand(S) %利用函数 expand() 进行符号表达式的展开
ans =
2*x^2+y^2
>> syms x y
expand(cos(x+y))
ans =
cos(x)*cos(y)-sin(x)*sin(y)
```

3. 嵌套格式

函数 `horner()` 用于将符号表达式化为嵌套格式, 嵌套格式可提高算法的效率。其调用格式如下。

`R = horner(P)`: 其中 `P` 为待嵌套的符号表达式, `R` 为嵌套后的符号表达式。

【例 8.10】 嵌套格式的生成。

```
>> S=sym('x*(2*x+y)+y*(x+y)');
horner(S)
ans =
y^2+(2*y+2*x)*x
```

4. 因式分解

函数 `factor()` 可用于对符号表达式进行因式分解, 同时也可对某一整数进行因式分解, 其调用的格式如下。

`f = factor(n)`: `n` 为待分解的表达式或正整数, `f` 为分解后的结果。

【例 8.11】 因式分解。

```
>> S=sym('x^3+2*x+3');
factor(S) %符号表达式的因式分解
```



```
ans =
(x+1)*(x^2-x+3)
>> factor(33)           %正整数的因式分解
ans =
3      11
```

8.3.3 符号表达式的代数运算

符号表达式的代数运算与数值运算类似，也可以通过“+”、“-”、“*”、“/”、“^”等运算符来实现。同时，相应的函数 `symadd()`、`symsub()`、`symmul()`、`symdiv()`和 `sympow()`分别用于实现符号表达式的加、减、乘、除、幂运算。建议读者还是以运算符的方式实现符号表达式的代数运算较为简单，笔者不推荐使用函数的形式计算，感兴趣的读者可自行阅读相关的帮助文档。下面以一具体实例讲解符号表达式的代数运算。

【例 8.12】符号表达式的代数运算。

```
>> f = sym('x-2');
g = sym('x^2+x-6');
>> z1 = f+g           %符号表达式的加法运算
z1 =
2*x-8+x^2
>> z2 = f-g           %符号表达式的减法运算
z2 =
4-x^2
>> z3 = f*g           %符号表达式的乘法运算
z3 =
(x-2)*(x^2+x-6)
>> z4 = f/g           %符号表达式的除法运算
z4 =
(x-2)/(x^2+x-6)
>> z5 = f^2           %符号表达式的幂运算
z5 =
(x-2)^2
```

8.3.4 符号表达式的分子分母提取

当符号表达式为有理分式时，函数 `numden()`可用于提取有理分式的分子和分母，其调用格式如下。

`[N,D] = numden(A)`: 输入参数 `A` 为符号表达式，返回参数 `N` 和 `D` 分别为符号参数 `A` 的分子和分母。

【例 8.13】符号表达式的分子分母提取。

```
>> A=sym('(x-1)/(b+x)');
[D,N]=numden(A)
D =
x-1           %符号表达式的分子
N =
b+x           %符号表达式的分母
```

8.3.5 符号表达式的自变量的确定

符号表达式往往会有多个符号变量，而在求解微积分、解方程等复杂数学问题中需要有明确的自变量，默认状态下，MATLAB 对自变量的确定机制如下。

- 自变量通常为小写字母，一般为 `x`、`y`、`z`、`t` 等。
- 小写字母 `i` 和 `j` 不作为自变量。
- 优先选择 `x` 为自变量，如果不存在 `x`，则按字母表的顺序搜索与 `x` 接近的字符自变量。

函数 `findsym()` 可用于确定运算过程中的符号自变量，其调用格式如下。

- `r = findsym(S)`: 给出符号表达式 `S` 中的所有符号变量。
- `r = findsym(S,n)`: 按照数学习惯中符号变量的优先顺序，确定符号表达式中的 `n` 个符号变量。

【例 8.14】 符号变量的确定。

```
>> syms x y z t w           %符号变量定义
f=x*y+2*z+t*(w+t);
>> r = findsym(f)           %查找符号表达式中的所有符号变量
r =
t, w, x, y, z
>> r1= findsym(f,1)         %查找符号表达式中的默认符号变量
r1 =
x
>> r2= findsym(f,2)
r2 =
x,y
>> r3= findsym(f,3)
r3 =
x,y,w
>> r4= findsym(f,4)
r4 =
x,y,w,z
```

8.4 符号矩阵运算

矩阵运算体现了 MATLAB 处理科学计算问题的强大功能，相应的符号矩阵运算也是我们处理、分析实际工程问题的重要手段。符号矩阵的运算类似于数值矩阵，包括了代数运算和一些特殊的矩阵操作，本节将向读者介绍符号矩阵运算的基本知识。

8.4.1 符号矩阵的代数运算

符号矩阵的代数运算包括一般的加、减、乘、除等四则运算。符号矩阵的代数运算是把矩阵当做一个整体，按照代数运算的准则进行运算，其中的运算基本同数值矩阵的运算规则，下面主要以实例来讲解符号矩阵的代数运算。

【例 8.15】 符号矩阵的代数运算。

```
>> A=sym(' [x^2+y x;x-y y^2] ') %创建符号矩阵
A =
[ x^2+y,      x]
[  x-y,      y^2]
>> B=sym(' [x+3*y x/3;x+y y+2] ')
B =
[ x+3*y,      x/3]
[  x+y,      y+2]
>> C=A+B           %符号矩阵的加法
C =
[ x^2+4*y+x,      4/3*x]
[      2*x,      y^2+y+2]
>> C=A*B           %符号矩阵的乘法
C =
[ (x^2+y)*(x+3*y)+x*(x+y),      1/3*(x^2+y)*x+x*(y+2) ]
[ (x-y)*(x+3*y)+y^2*(x+y),      1/3*(x-y)*x+y^2*(y+2) ]
>> C=A./B          %符号矩阵的点除
C =
```

```

[ (x^2+y)/(x+3*y),          3]
[ (x-y)/(x+y),          y^2/(y+2) ]
>> C=A/B %符号矩阵除法
C =
[-3*(y*x^2+y^2-x*y+x^2+2*y)/(x^2-2*x*y-6*x-9*y^2-18*y),
x*(x^2-8*y-3*x)/(x^2-2*x*y-6*x-9*y^2-18*y)]
[3*(-x*y+y^2+y^3-2*x+2*y+x*y^2)/(x^2-2*x*y-6*x-9*y^2-18*y),
(x^2-x*y-3*x*y^2-9*y^3)/(x^2-2*x*y-6*x-9*y^2-18*y)]
>> C=A.^2 %符号矩阵点乘方
C =
[ (x^2+y)^2,          x^2]
[ (x-y)^2,          y^4]
>> C=A^2 %符号矩阵乘方
C =
[ (x^2+y)^2+(x-y)*x,          (x^2+y)*x+x*y^2]
[ (x-y)*(x^2+y)+y^2*(x-y),          (x-y)*x+y^4]

```

8.4.2 符号矩阵的特殊运算

符号矩阵的特殊运算主要包括符号矩阵的转置、行列式、求逆、求秩、特征值、奇异值计算，这部分的计算与数值矩阵的用法基本相同，本节以实例演示符号矩阵的这些运算。

【例 8.16】符号矩阵的特殊运算。

(1) 创建符号矩阵。

```

>> x=sym('[1/3 1/4; 3/7 2/9]')
x =
[ 1/3, 1/4]
[ 3/7, 2/9]

```

(2) 符号矩阵的转置。

```

>> transpose(x)
ans =
[ 1/3, 3/7]
[ 1/4, 2/9]

```

(3) 符号矩阵的行列式。

```

>> det(x)
ans =
-25/756

```

(4) 符号矩阵求逆。

```

>> inv(x)
ans =
[ -168/25, 189/25]
[ 324/25, -252/25]

```

(5) 符号矩阵求秩。

```

>> rank(x)
ans =
2

```

(6) 符号矩阵特征值计算。

```

>> [m,n]=eig(x)
m =
[ 1, 1]
[ -2/9+10/63*70^(1/2), -2/9-10/63*70^(1/2) ]
n =
[ 5/18+5/126*70^(1/2), 0]
[ 0, 5/18-5/126*70^(1/2) ]

```

(7) 符号矩阵奇异值计算。

```

>> svd(x)
ans =
5/504*1201^(1/2)+5/504*865^(1/2)

```

```
5/504*1201^(1/2)-5/504*865^(1/2)
```

8.5 符号微积分运算

微积分运算是工程领域经常会遇到的问题，MATLAB 符号工具箱可以较好地解决符号微积分的问题，下面从符号极限、符号微分、符号积分和符号级数 4 个方面介绍符号微积分在 MATLAB 中的运算。

8.5.1 符号极限

在 MATLAB 中，计算符号极限的函数为 `limit()`，其调用格式如下。

- `limit(F,x,a)`：计算符号表达式 F 在 $x \rightarrow a$ 时的极限值。
- `limit(F,a)`：计算符号表达式 F 默认符号自变量趋于 a 时的极限值。
- `limit(F)`：计算符号表达式 F 默认符号自变量趋于 0 时的极限值。
- `limit(F,x,a,'right')`：计算符号表达式 F 在 $x \rightarrow a$ 时的右极限值。
- `limit(F,x,a,'left')`：计算符号表达式 F 在 $x \rightarrow a$ 时的左极限值。

【例 8.17】 符号极限的计算。

```
>> F1=sym('cos(x)');  
limit(F1,x,0)                                %计算符号函数 x→0 时的极限  
ans =  
1  
>> F2=sym('1/sin(x)');  
limit(F2,x,0,'right')                        %计算符号函数 x→0 时的右极限  
ans =  
Inf  
>> limit(F2,x,0,'left')                      %计算符号函数 x→0 时的左极限  
ans =  
-Inf  
>> F3=sym('(x+y-2*z)/(x-y+2*z)');  
limit(F3)  
ans =  
-1
```

8.5.2 符号微分

函数 `diff()` 可用于实现符号微分，其调用格式如下。

- `Y = diff(F)`：对符号函数 F 的默认符号变量进行一阶微分。
- `Y = diff(F,'t')`：对符号函数 F 的符号变量 t 进行一阶微分。
- `Y = diff(F,'t',n)`：对符号函数 F 的符号变量 t 进行 n 阶微分。

【例 8.18】 符号微分的计算。

```
>> F=sym('cos(x)+sin(y)');  
Y=diff(F)                                    %对符号表达式 F 默认的符号变量 x 进行一阶微分  
Y =  
-sin(x)  
>> Y=diff(F,'y')                            %对符号表达式 F 的符号变量 y 进行一阶微分  
Y =  
cos(y)  
>> Y=diff(F,'y',2)                          %对符号表达式 F 的符号变量 y 进行二阶微分  
Y =  
-sin(y)  
>> f=sym('[x sin(x);1/5 1/y]')             %创建符号矩阵
```

```
f =
[      x, sin(x)]
[  1/5,   1/y]
>> y=diff(f) %对符号矩阵进行微分，即对矩阵各元素分别进行微分
y =
[      1, cos(x)]
[      0,      0]
```

函数 `jacobian()` 可用于多元符号函数的微分计算。函数 `jacobian()` 用于计算 jacobian 矩阵，其调用格式如下。

`R = jacobian(f,v)`: 用于计算函数向量 `f` 对自变量向量 `x` 的 jacobian 矩阵。

【例 8.19】多元符号函数的微分计算。

```
>> syms x y
f=sym('[x sin(x+y);1/5 x/y]')
R = jacobian(f,[x,y]) %计算 jacobian 矩阵
f =
[      x, sin(x+y)]
[  1/5,   x/y]
R =
[      1,      0]
[      0,      0]
[ cos(x+y), cos(x+y)]
[  1/y,   -x/y^2]
```

8.5.3 符号积分

函数 `int()` 可用于符号积分，其调用格式如下。

- `R = int(S)`: 计算符号函数 `S` 对默认的自变量符号的不定积分。
- `R = int(S,v)`: 计算符号函数 `S` 对自变量符号 `v` 的不定积分。
- `R = int(S,a,b)`: 计算符号函数 `S` 对默认的自变量符号在 `[a,b]` 上的定积分值。
- `R = int(S,v,a,b)`: 计算符号函数 `S` 对自变量符号 `v` 在 `[a,b]` 上的定积分值。

【例 8.20】符号积分的计算。

```
>> F1=sym('cos(x)');
r1=int(F1) %不定积分计算
r1 =
sin(x)
>>F2=sym('cos(x)');
r2=int(F2,1,3) %定积分计算
r2 =
sin(3)-sin(1)
```

8.5.4 符号级数

在 MATLAB 中符号级数常用函数 `symsum()` 和 `taylor()` 来计算。其中，函数 `symsum()` 用于对级数求和，其调用格式如下。

- `r = symsum(s)`: 计算符号表达式 `s` 对默认的自变量符号 `v` 在 `[0,v-1]` 范围内的级数和。
- `r = symsum(s,v)`: 计算符号表达式 `s` 对自变量符号 `v` 在 `[0,v-1]` 范围内的级数和。
- `r = symsum(s,a,b)`: 计算符号表达式 `s` 对默认的自变量符号在 `[a,b]` 范围内的级数和。
- `r = symsum(s,v,a,b)`: 计算符号表达式 `s` 对自变量符号 `v` 在 `[a,b]` 范围内的级数和。

【例 8.21】符号级数求和的计算。

```
>> s=sym('sin(x)'); %计算符号函数的级数和
r1 = symsum(s)
r1 =
1/2*sin(1)/(cos(1)-1)*cos(x)-1/2*sin(x)
>> r2 = symsum(s,1,3)
```

```
r2 =
sin(1)+sin(2)+sin(3)
```

函数 `taylor()` 可用于计算泰勒级数展开的 n 项，函数的调用格式如下。

- `r = taylor(f)`: 计算符号函数 f 的泰勒级数展开项，默认的展开项为 5 项，自变量为默认。
- `r = taylor(f,n,v)`: 计算自变量为 v 的 n 阶泰勒级数展开项。

【例 8.22】 符号泰勒级数展开的计算。

```
>> sym x
r1=taylor(sin(x)) %计算默认状态下的符号函数的泰勒级数展开
ans =
x
r1 =
x-1/6*x^3+1/120*x^5
>> r2=taylor(sin(x),10) %计算符号函数的 10 阶的泰勒级数展开
r2 =
x-1/6*x^3+1/120*x^5-1/5040*x^7+1/362880*x^9
```

8.6 符号方程求解

方程求解是科学计算领域经常遇到的问题，前面的章节中介绍了方程的数值求解方法，本章将介绍方程的符号求解。在本节中关于符号方程的求解将从代数方程和微分方程两个方面讲解，通过本节的学习读者将掌握符号方程求解的能力。

8.6.1 代数方程的求解

在符号工具箱中，函数 `solve()` 用于求解代数方程和方程组，其调用格式如下。

- `g = solve(eq)`: 计算方程 eq 的解，方程 eq 可为字符串或符号表达式，返回的结果是方程 eq 默认字符自变量的解。
- `g = solve(eq,var)`: 计算方程 eq 的解，对方程中的字符变量 var 求解。
- `g = solve(eq1,eq2,...,eqn)`: 求解符号方程组 $eq1, eq2, \dots, eqn$ 。
- `g = solve(eq1,eq2,...,eqn,var1,var2,...,varn)`: 求解符号方程组 $eq1, eq2, \dots, eqn$ ，并指定各符号方程的符号自变量 $var1, var2, \dots, varn$ 。

【例 8.23】 代数方程的求解。

(1) 求解代数方程: $3x^2+5x+3=0$ 。

```
>> y=solve('3*x^2+5*x+3')
y =
-5/6+1/6*i*11^(1/2)
-5/6-1/6*i*11^(1/2)
```

(2) 求解代数方程组:

$$\begin{cases} x+y=5 \\ x-2y=7 \end{cases}$$

```
>> [x,y]=solve('x+y-5','x-2*y-7')
x =
17/3
y =
-2/3
```

8.6.2 微分方程的求解

MATLAB 中函数 `dsolve()` 用于微分方程的求解，其调用格式如下。

- `r = dsolve('eq1,eq2,...','cond1,cond2,...','v')`: 求解微分方程 `eq1, eq2...`，参数 `cond1`、`cond2` 为微分方程的初始条件，`v` 为指定求解的符号自变量，默认为 `t`，其中微分方程 `eq` 的书写格式为 `D` 表示微分，默认符号变量 `t` 下，`Dy` 表示 dy/dt ，`D2y` 表示 d^2y/dt^2 。
- `r = dsolve('eq1','eq2',...,'cond1','cond2',...,'v')`: 此格式与上面的格式仅在字符串符号的位置上有所区别。

【例 8.24】 微分方程的求解。

求解微分方程：

$$\frac{d^2y}{dx^2} - 3\frac{dy}{dx} + y = 0$$

```
>> y=dsolve('D2y-3Dy+y=0')
y =
sin(t)*C2+cos(t)*C1+3
```

8.7 符号函数图形绘制

MATLAB 中提供了符号绘图函数可将符号函数快速地以图形化的方式显示，便于用户全面、直观地了解函数。利用符号函数绘制图形，在前面的图形绘制相关章节已做了简单介绍，这里将更为详细、全面地向读者介绍符号函数图形绘制的内容。

8.7.1 函数 `ezplot()`

函数 `ezplot()` 用于绘制单变量或者两个变量的符号函数图，绘制函数 $y=f(x)$ 或 $z=f(x,y)$ 的二维图形。其调用格式如下。

- `ezplot(f)`: 绘制符号函数 f 的二维图。
- `ezplot(f,[min,max])`: 绘制符号函数 $y=f(x)$ 的二维图，并设置绘图的数据范围。
- `ezplot(f,[xmin,xmax,ymin,ymax])`: 绘制符号函数 $z=f(x,y)$ 的二维图，并设置绘图的数据范围。

【例 8.25】 利用函数 `ezplot()` 绘制符号二维曲线图。

```
ezplot('x^2-y^4')
```

执行上述程序，将生成如图 8.1 所示的二维曲线图。

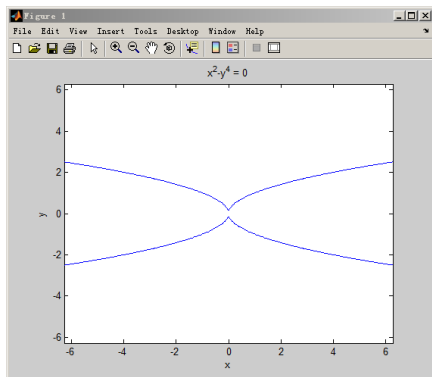


图 8.1 符号函数 `ezplot()` 绘制的二维曲线图

8.7.2 函数 fplot()

函数 `fplot()` 可绘制与函数 `plot()` 类似的二维图，用于反映自变量与因变量的关系，所不同的是使用符号函数绘图时无须设置图形坐标的范围。函数 `fplot()` 的调用格式如下。

- `fplot(function,limits)`: 快速绘制符号函数 `function` 的二维曲线图，参数 `limits` 用于设置绘图坐标轴的范围，为数据向量 `[xmin xmax ymin ymax]`。
- `fplot(function,limits,LineSpec)`: 快速绘制符号函数 `function` 的二维曲线图，并通过参数设置 `LineSpec` 其线性。

【例 8.26】 利用函数 `fplot()` 绘制符号二维曲线图。

```
fplot('sin(x)', [-2*pi, 2*pi])
```

执行上述程序，将生成如图 8.2 所示的二维曲线图。

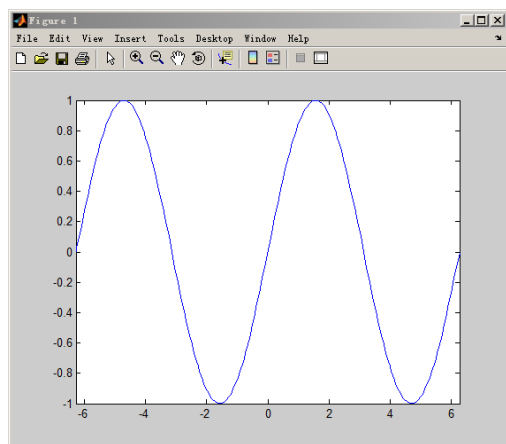


图 8.2 符号函数 `fplot()` 绘制的二维曲线图

8.7.3 函数 ezplot3()

函数 `ezplot3()` 可用于绘制三维曲线图，其基本的调用格式如下。

`ezplot3(x,y,z)`: 绘制数据 `x`、`y`、`z` 的三维曲线图。

【例 8.27】 利用函数 `ezplot3()` 绘制符号三维曲线图。

```
ezplot3('cos(x)', '2*x', 'sin(x)')
```

执行上述程序，将生成如图 8.3 所示的三维曲线图。

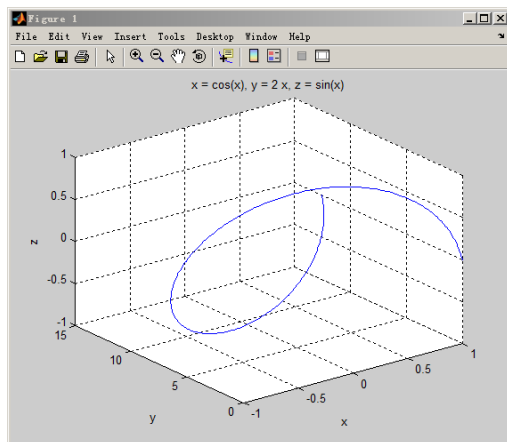


图 8.3 符号函数 `ezplot3()` 绘制的三维曲线图

8.7.4 函数 ezcontour()

函数 ezcontour() 可用于绘制符号函数的等高线图，其调用格式如下。

- ezcontour(f): 绘制符号函数 f 的等高线图，默认的定义域为 $[-2\pi, 2\pi]$ 。
- ezcontour(f, domain): 在指定的数据域内绘制符号函数 f 的等高线图。
- ezcontour(..., n): 在 $n \times n$ 的网格内绘制等高线图。
- ezcontour(axes_handle, ...): 绘制符号函数的等高线图，并设置其坐标轴属性。
- $h = \text{ezcontour}(\dots)$: 绘制符号函数的等高线图，并返回其句柄 h 。

【例 8.28】 利用函数 ezcontour() 绘制符号等高线图。

```
f=sym('x^2+y^2');
ezcontour(f)
```

执行上述程序，将生成如图 8.4 所示的等高线图。

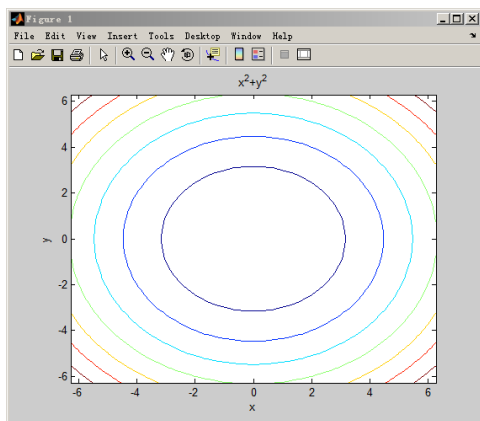


图 8.4 符号函数等高线图

8.7.5 函数 ezcontourf()

函数 ezcontourf() 用于绘制带颜色填充的等高线图，其调用格式与函数 ezcontour() 的类似，这里不详细展开叙述。

【例 8.29】 利用函数 ezcontourf() 绘制符号带颜色填充的等高线图。

```
f=sym('x^2+y^2');
ezcontourf(f)
```

执行上述程序，将生成如图 8.5 所示的带颜色填充的等高线图。

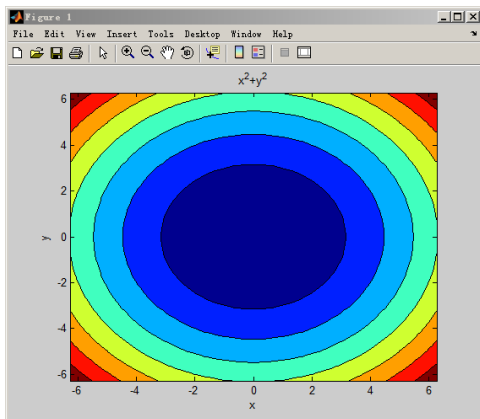


图 8.5 符号函数 ezcontourf()绘制带颜色填充的等高线图

8.7.6 函数 ezmesh()

函数 ezmesh()可用于绘制符号函数的三维网格图,其调用格式如下。

- ezmesh(f): 绘制符号函数 f 的三维网格图,默认的定义域为 $[-2\pi, 2\pi]$ 。
- ezmesh(f,domain): 在指定的数据域 domain 内,绘制符号函数 f 的三维网格图。
- ezmesh(x,y,z): 绘制由 $x=x(s,t)$, $y=y(s,t)$, $z=z(s,t)$ 组成的符号函数 $z=f(x,y)$ 的三维网格图。
- ezmesh(x,y,z,[smin,smax,tmin,tmax]) or ezmesh(x,y,z,[min,max]): 绘制三维网格图,并设置数据范围。
- ezmesh(...,n): 在 $n \times n$ 的网格内绘制符号函数的三维网格图。
- ezmesh(...,'circ'): 在圆形区域内绘制符号函数的三维网格图。
- ezmesh(axes_handle,...): 设置符号函数的三维网格图的坐标轴属性。
- h = ezmesh(...): 返回符号函数的三维网格图的句柄 h。

【例 8.30】利用函数 ezmesh()绘制符号三维网格图。

```
f=sym('x^2+y^2');
ezmesh(f)
```

执行上述程序,将生成如图 8.6 所示的三维网格图。

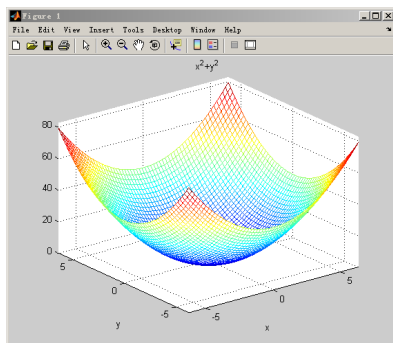


图 8.6 符号函数 ezmesh()绘制三维网格图

8.7.7 函数 ezmeshc()

函数 ezmeshc()也可用于绘制符号函数的三维网格图,并在 xy 轴平面上投影网格图的等高线。其调用格式与函数 ezmesh()类似,下面以一实例来演示该函数的用法。

【例 8.31】利用函数 ezmeshc()绘制等高线投影的符号三维网格图。

```
f=sym('x^2+y^2');
ezmeshc(f)
```

执行上述程序,将生成如图 8.7 所示的带等高线投影的三维网格图。

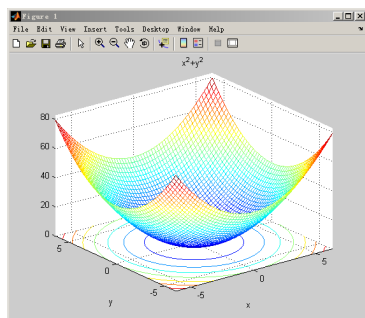


图 8.7 符号函数 ezmeshc()绘制带等高线投影的三维网格图

另外，函数 `ezsurf()` 和 `ezsurf()` 可用于绘制三维曲面图，其使用方法与函数 `ezmesh()` 和 `ezmeshc()` 类似，这里不再详细展开叙述。

8.7.8 函数 `ezpolar()`

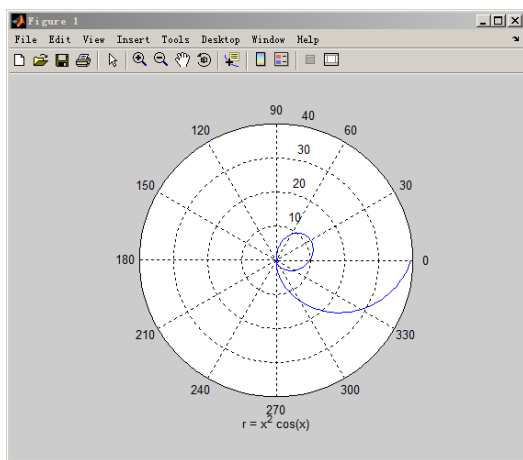
函数 `ezpolar()` 可用于绘制符号函数的极坐标图，其调用格式如下。

- `ezpolar(f)`: 绘制符号函数 f 的极坐标图。
- `ezpolar(f,[a,b])`: 绘制指定数据范围 $[a,b]$ 的符号函数 f 的极坐标图。

【例 8.32】 利用函数 `ezpolar()` 绘制符号极坐标图。

```
ezpolar('x^2*cos(x)')
```

执行上述程序，将生成如图 8.8 所示的极坐标图。

图 8.8 符号函数 `ezpolar()` 绘制的极坐标图

8.8 符号对象与数值对象的转换

尽管 MATLAB 为用户提供了强大的符号计算功能，但很多时候为了与其他程序算法有更好的融合，需要将符号对象转换为数值对象。同时，相应的一些数值对象无法求解的问题也需要转换为符号对象求解。本节主要向读者介绍如何在 MATLAB 中实现符号对象和数值对象的转换，通过本节的学习，读者将熟练掌握转换的方法，在今后的使用中符号对象与数值对象将可以更好地交互应用。

8.8.1 符号对象转换为数值对象

函数 `subs()` 可将符号计算中的符号变量以数值来代替，可以将指定的一个或多个符号变量替换为数值，并返回计算结果。函数 `subs()` 调用格式如下。

- $R = \text{subs}(S)$: 将符号表达式中的所有符号变量用数值代替，未指定代替的具体值，函数会自动搜索 MATLAB 的工作空间，寻找相应的值替代符号变量，如果未找到相应的数值变量，则符号变量保持不变。
- $R = \text{subs}(S, \text{new})$: 指定符号变量替代的新值 `new`，MATLAB 会首先置换其默认的符号自变量，参见本章符号表达式的自变量的确定的内容。

- `R = subs(S,old,new)`: 指定需要替换的符号变量 `old` 及替换后的值 `new`。

【例 8.33】符号变量的数值替换。

```
>> f1=sym('x*x+sin(x)')           %创建符号表达式
f1 =
x*x+sin(x)
>> subs(f1,2)                     %符号变量 x 赋值为 2
ans =
4.9093
>> f2=sym('x*x+sin(y)')
f2 =
x*x+sin(y)
>> subs(f2,2)                     %对符号表达式的默认自变量符号赋值
ans =
4+sin(y)
>> subs(f2,'y',2)                 %对符号表达式的符号变量 y 使用数值替换
ans =
x^2+sin(2)
>> syms a b c
f3=a.^2+(b-c)^3;
subs(f3,{b,c},{2,3})              %对符号表达式的多个符号变量赋值
ans =
a^2-1
>> subs(f2,findsym(f2),{1,2,3})  %对符号表达式的所有符号变量赋值
ans =
x*x+sin(y)
```

8.8.2 数值对象转换为符号对象

函数 `sym()`除了具有符号对象的创建功能外,还可以将数值对象转换为符号对象,其中的输入参数即为待转换的数值变量,参数 `flag` 可以设置转换后的数据精度。

【例 8.34】数值变量的符号替换。

```
>> a=0.75;
A=sym(a)                           %数值变量转换为符号变量
class(A)
A =
3/4
ans =
sym
>> b=magic(3)./5;                  %数值矩阵转换为符号矩阵
B =
[ 8/5, 1/5, 6/5]
[ 3/5, 1, 7/5]
[ 4/5, 9/5, 2/5]
```

8.9 本章小结

本章主要介绍了如何利用 MATLAB 进行符号计算,通过本章的学习,读者应该掌握以下内容:符号对象的创建、符号表达式的基本操作、符号矩阵的代数和特殊运算、符号极限、符号微分、符号积分、符号级数、符号方程求解、符号函数图形绘制、符号对象与数值对象的转换。对于这些内容的学习将有助于读者解决算法推导与分析的问题。

本章主要讲述了符号计算的相关内容,但并不是所有的问题都可以利用符号计算来求解的,读者应根据自己的实际问题选择符号计算或者数值计算功能求解。



第9章

应用程序接口

MATLAB 在数值分析、图像处理等各方面都体现出了卓越的性能,但是, MATLAB 是边解释边执行的程序语言,导致其程序运行较慢,在实际应用中可执行文件更受用户的欢迎。同时, MATLAB 程序的运行又不可以脱离 MATLAB 环境,近年来,随着 MATLAB 软件功能的不断完善, MATLAB 软件也越来越大,用户希望能够脱离 MATLAB 庞大的编程环境,独立运行程序。再者,一些大型的软件往往基于 C/C++ 平台开发,虽然这些高级语言可生成独立的可执行文件且执行效率高,但是在开发过程中程序员又希望可以调用 MATLAB 的一些复杂算法,提高软件设计的效率。为此, MathWorks 公司提供了 MATLAB 编译器、MEX 文件、MAT 文件、MATLAB 引擎技术等,用于应用程序的接口编程。

本章将主要介绍常用的接口编程技术,包括 MATLAB 编辑器的简介、安装、使用,由其他高级语言生成的可供 MATLAB 调用的 MEX 文件,实现不同程序接口间数据传递的 MAT 文件,在其他语言中调用 MATLAB 函数的引擎技术,基于 COM 组件的接口编程技术, MATLAB 与 Word、Excel 的混合使用。

9.1 MATLAB 编译器

MATLAB 编译器用于 MATLAB 环境下的编译工作。MATLAB 编译器可将 M 文件生成独立的可执行文件、函数库 dll 文件、组件 (COM 等),可脱离 MATLAB 环境直接运行或者供其他高级语言调用,不受应用平台的限制。

9.1.1 编译器简介

MATLAB 编译器的工作原理是将 M 文件转换成 C/C++ 代码,然后再调用配置好的 C/C++ 编译器把产生的源代码编译成用户指定的格式,包括如下几种。

- 可执行文件 (exe): 面向用户的直接使用的应用程序,可脱离 MATLAB 软件环境的支持。
- C/C++ 语言的动态链接库函数: 可整合到 C/C++ 应用程序中,与 C/C++ 应用程序一同编译、执行,实现不同程序语言间的连接。
- 程序组件: 可供其他外部程序调用。

MATLAB 编译器由 3 个部分构成,即 MATLAB Compiler、MATLAB Component Runtime (MCR) 和 Component Technology File (CTF)。其中: MATLAB Compiler 是 MATLAB 专用的编译器,可编译 C/C++ 源代码; MCR 是 MATLAB Component Runtime 的缩写,它是一组独立的函数共享库,在应用程序发布时一同提供给用户,用户通过它能够执行在 MATLAB 中编写的 M 文件; CTF 是 Component Technology File 的缩写,是一种压缩技术,通过它 MATLAB 将可部署文件封装起来。

使用 MATLAB 编译器可以使其他外部程序调用 MATLAB 的相应功能；可以生成独立的程序文件；可以提高程序的效率；可以封装 M 文件。但是 MATLAB 编译器目前来说还无法对 MATLAB 所有的函数和工具箱进行编译，其中符号工具箱、神经网络工具箱、图形界面相关的一些工具箱、命令行直接调用的函数无法完成编译工作。

9.1.2 编译器的安装、配置

MATLAB 编译器的安装包括 MATLAB Compiler 和 C/C++ 编译器，其中 MATLAB Compiler 在安装 MATLAB 软件的时候默认状态下都会自动安装上，而 C/C++ 编译器需要再安装，并对其进行一定的配置。C/C++ 编译器用于将 MATLAB Compiler 编译产生的代码生成用户需要的接口程序，MATLAB 支持的 C/C++ 编译器主要有如下 3 个。

- MATLAB 本身自带的 Lcc 编译器：不需要外部安装，只需要进行一定配置即可，但是该编译器只能编译 C 代码，而不能编译 C++ 代码。
- Borland C++：需要外部安装，可编译 C/C++ 代码，可以安装的版本包括 5.3、5.4、5.6。
- Microsoft Visual C++：需要外部安装，可编译 C/C++ 代码，可以安装的版本包括 6.0、7.0、7.1 及 8.0。

在确定 MATLAB 编译器安装完成后，还需要对其进行一定的配置工作，包括：

(1) 在命令行输入如下命令，启动编译器的配置工作。

```
mex -setup
```

(2) 系统会提示如下信息，提示用户选择需要配置的编译器，用户选择“y”可以让 MATLAB 自动搜索可以利用的编译器。

```
Please choose your compiler for building external interface (MEX) files:
Would you like mex to locate installed compilers [y]/n?
```

(3) 在上一步的操作中选择了让系统自动搜索可供配置的编译器，系统中将显示编译器的类型和位置，以供用户选择需要配置的编译器。

```
Select a compiler:
[1] Lcc C version 2.4 in C:\MATLAB701\sys\lcc
[2] Microsoft Visual C/C++ version 6.0 in C:\Program Files (x86)\Microsoft Visual Studio
[0] None
Compiler:
```

(4) 选择了待配置的编译器后，MATLAB 会给出如下所示的提示信息，向用户验证是否确定当前选择的编译器。

```
Please verify your choices:
Compiler: Microsoft Visual C/C++ 6.0
Location: C:\Program Files (x86)\Microsoft Visual Studio
Are these correct?([y]/n):
```

(5) 输入“y”确定配置工作，MATLAB 对编译器进行相关配置后，将给出如下信息。

```
Try to update options file: C:\Users\ww\Application Data\MathWorks\
MATLAB\R14\mexopts.bat
From template: C:\MATLAB701\BIN\WIN32\mexopts\msvc60opts.bat
Done . . .
```

(6) 编译器的配置工作基本完成。配置完成后，下次使用 MATLAB 时，编译器的配置仍然有效。如果用户下次需要使用其他的编译器时，需要对系统的编译器环境进行重配置，其命令如下。

```
>> mbuild -setup
```

(7) 在命令窗口输入如下命令后，将显示如下所示信息，选择配置的编译器，操作回到第一次配置编译器的环节，之后的步骤也类似。

```
Please choose your compiler for building standalone MATLAB applications:
Would you like mbuild to locate installed compilers [y]/n?
```

关于编译器的选择，建议用户最好采用 Microsoft Visual C/C++ 6.0，该编译器性能较为稳定，

9.1.3 编译器的使用

MATLAB 中提供了函数 `mcc()` 实现编译工作, 本小节中将主要介绍函数 `mcc()` 的使用。

`mcc()` 函数将 M 文件编译为多种应用程序类型, 其调用格式如下。

`mcc [-options] mfile1 [mfile2 ... mfileN]`: 用于编译 M 文件, 其中参数 `options` 设置编译后的文件类型, 可选类型包括 `'-x'` (生成 MEX 文件, 即 dll 文件)、`'-m'` (生成独立执行的 exe 文件)、`'-p'` (生成独立执行的 C++ 文件)、`'-s'` (生成 simulink S 函数)、`'-B'` (生成可独立运行的 C 图形库函数)、`'-B sgl'` (生成可独立运行的 C++ 图形库函数)、`'-m -W'` (生成 C 函数库)。

Matlab 编译器使用 `mcc()` 命令将 M 文件翻译成 C/C++ 程序文件, 同时生成一个 CTF 格式的 MATLAB 压缩文件, 该文件包含了编译器产生的代码和其支持的可执行文件类型之间的接口。利用 CTF 文件提供的接口生成的 C 文件能用于各种可执行文件的函数类型 (MEX 文件、EXE 文件、DLL 文件)。

在使用编译器的时候需要注意编译的 M 文件必须为函数文件, 因此如果是脚本文件需要编译, 必须改写脚本文件为函数文件。最简单的方法是在脚本文件上方添加函数声明行。

【例 9.1】编译器的使用。

```
mcc -m myfun
```

将 M 文件 `filename.m` 翻译成 C 代码, 生成的可执行文件 (exe) 能独立于 MATLAB 运行环境。运行可执行文件, 结果将在 DOS 界面中生成, 如图 9.1 所示。

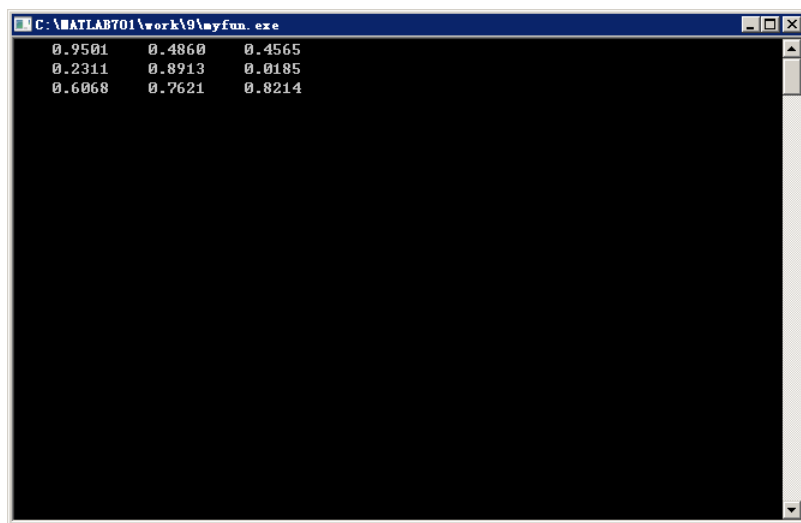


图 9.1 可执行文件的 DOS 结果显示

9.2 MEX 文件

MEX 文件是在 MATLAB 环境中调用的 C 或其他高级语言编译生成的文件, 即 MEX 文件的源代码是其他高级语言编写的, 在 MATLAB 中通过 MEX 的文件格式被调用。

在下面一些情况下可以考虑使用 MEX 文件。

- 当 MATLAB 中需要用到的一些函数功能已编写了现成的 C 或其他高级语言的代码, 在 MATLAB 中即可直接通过 MEX 文件的形式使用, 无须重新编写为 M 文件。
- MATLAB 是解释性的语言, 算法的执行效率可能比较慢, 特别是一些循环操作, 可以考虑

采用效率较高的 C 等高级语言编写，在 MATLAB 中调用其编译好的文件。

MEX 文件源代码是由 C 等高级语言编写的，通过 MATLAB 编译器编译为可执行的动态链接函数。在 Windows 系统中 MEX 文件的建立过程可以分为编译、预链接和链接 3 个过程。mex 命令编译的 C/C++ 源文件中不含 main() 函数，是以 mexfunction() 函数作为 Matlab 调用的入口。

MEX 文件的使用极为方便，调用格式为“MEX 文件名”，同时在 MATLAB 搜索机制中 MEX 文件的优先级高于 M 文件，即调用 MEX 文件的时候，如果有同名的 M 文件，MEX 文件会优先被调用。

下面以 MATLAB 自带的 MEX 应用示例文件 yprime 为例，向读者演示其使用方法。该示例通过 MATLAB 编译器将 C 语言编写的 yprime.c 文件编译为 MEX 文件。

【例 9.2】MEX 文件的使用。

函数 yprime() 的 C 源程序。

```
/*=====
 *
 * YPRIME.C Sample .MEX file corresponding to YPRIME.M
 *      Solves simple 3 body orbit problem
 *
 * The calling syntax is:函数的调用格式
 *
 *      [yp] = yprime(t, y)
 *
 * You may also want to look at the corresponding M-code, yprime.m.
 *
 * This is a MEX-file for MATLAB.
 * Copyright 1984-2000 The MathWorks, Inc.
 *
 *=====*/
/* $Revision: 1.10 $ */
#include <math.h>
#include "mex.h"
/* Input Arguments */
#define T_IN   prhs[0]
#define Y_IN   prhs[1]
/* Output Arguments */
#define YP_OUT plhs[0]
#ifdef MAX
#define MAX(A, B) ((A) > (B) ? (A) : (B))
#endif
#ifdef MIN
#define MIN(A, B) ((A) < (B) ? (A) : (B))
#endif
#define PI 3.14159265
/*以下为 yprime 文件的主函数*/
static double mu = 1/82.45;
static double mus = 1 - 1/82.45;
static void yprime(
    double yp[],
    double *t,
    double y[]
)
{
    double r1,r2;

    r1 = sqrt((y[0]+mu)*(y[0]+mu) + y[2]*y[2]);
    r2 = sqrt((y[0]-mus)*(y[0]-mus) + y[2]*y[2]);
    /* Print warning if dividing by zero. */
    if (r1 == 0.0 || r2 == 0.0 ){
```



```

    mexWarnMsgTxt("Division by zero!\n");
}

yp[0] = y[1];
yp[1] = 2*y[3]+y[0]-mus*(y[0]+mu)/(r1*r1*r1)-mu*(y[0]-mus)/(r2*r2*r2);
yp[2] = y[3];
yp[3] = -2*y[1] + y[2] - mus*y[2]/(r1*r1*r1) - mu*y[2]/(r2*r2*r2);
return;
}
/*以下为 MEX 文件的入口函数*/
void mexFunction( int nlhs, mxArray *plhs[],
                  int nrhs, const mxArray*prhs[] )
{
    double *yp;
    double *t,*y;
    unsigned int m,n;

    /* Check for proper number of arguments */

    if (nrhs != 2) {
        mexErrMsgTxt("Two input arguments required.");
    } else if (nlhs > 1) {
        mexErrMsgTxt("Too many output arguments.");
    }
    /* Check the dimensions of Y. Y can be 4 X 1 or 1 X 4. */
    m = mxGetM(Y_IN);
    n = mxGetN(Y_IN);
    if (!mxIsDouble(Y_IN) || mxIsComplex(Y_IN) ||
        (MAX(m,n) != 4) || (MIN(m,n) != 1)) {
        mexErrMsgTxt("YPRIME requires that Y be a 4 x 1 vector.");
    }
    /* Create a matrix for the return argument *创建输出的矩阵对象/
    YP_OUT = mxCreateDoubleMatrix(m, n, mxREAL);
    /* Assign pointers to the various parameters */
    yp = mxGetPr(YP_OUT);
    t = mxGetPr(T_IN);
    y = mxGetPr(Y_IN);
    /* Do the actual computations in a subroutine */
    yprime(yp,t,y);
    return;
}

```

在确定 MATLAB 编译器成功安装并配置好后，可以编译上述的 C 语言的源程序。在命令窗口输入如下命令。

```
mex yprime.c
```

编译完成，在当前目录下将生成文件“yprime.dll”。下面，在 MATLAB 中调用该动态链接函数。

```

>> [yp] = yprime(2,ones(4,1))
yp =
    1.0000
    2.6527
    1.0000
   -1.3551

```

9.3 MAT 文件

MAT 文件是 MATLAB 默认的数据存储形式，可存储一个或多个变量，用于向 MATLAB 中导

入和导出数据。MATLAB 工作空间内的数据可以通过 MAT 文件的形式导出到外界磁盘上，同时导出到外界磁盘上的 MAT 文件也可以再次导入 MATLAB 或其他程序语言中。

在 MATLAB 7.0 中通过函数 `save()` 保存工作空间内的数据，生成 MAT 格式的数据文件，在前面章节中对 `save()` 函数已进行了一定的介绍，这里再做一个简单的总结。

- `save`: 将当前 MATLAB 工作空间内的所有数据存储到名为 `matlab` 的 MAT 文件中。
- `save filename`: 将当前工作空间的所有数据存储到名为 `filename` 的 MAT 文件中。
- `save filename x y z`: 将当前工作空间的变量 `x`、`y`、`z` 存储到名为 `filename` 的 MAT 文件中。
- `save(..., 'format')`: 以指定的格式存储数据，参数 `format` 的可选项包括 “-append”（在原 MAT 文件中添加新的数据）、“-ascii”（数据保存为 8 位精度的 ASCII 文件）、“-ascii -double16”（数据保存为 16 位精度的 ASCII 文件）、“-ascii -tabs”（数据保存为用制表符间隔的 8 位精度的 ASCII 文件）、“-ascii -double -tabs”（数据保存为用制表符间隔的 16 位精度的 ASCII 文件）和 “-mat”（数据保存为二进制的 MAT 文件，默认选项）。

`load()` 函数可以从 MAT 文件中读取数据，其调用格式如下。

- `load`: 将 `matlab.mat` 中的数据导入当前 MATLAB 工作空间内。
- `load('filename')`: 导入文件名为 `filename` 的 MAT 文件内的数据。
- `load('filename', 'X', 'Y', 'Z')`: 导入文件名为 `filename` 的 MAT 文件内的数据 `X`、`Y`、`Z`。
- `load('-ascii', 'filename')`: 以文本的形式导入文件名为 `filename` 的 MAT 文件内的数据。

同时，MAT 文件可用于各种数据平台。为方便在不同的环境下使用 MAT 数据文件，MATLAB 提供了 MAT 文件函数库用于在不同的程序语言中操作 MAT 文件。MAT 文件函数库包含了各种对 MAT 文件进行读/写的函数，以 “mat” 为函数的前缀。

其中，在 C 语言中的读/写 MAT 文件的接口函数如下。

- 函数 `matOpen()`: 用于打开 MAT 文件。
- 函数 `matClose()`: 用于关闭 MAT 文件。
- 函数 `matGetDir()`: 从 MAT 文件中获得 MATLAB 数据的列表。
- 函数 `matGetFp()`: 获得指向 MAT 文件的指针。
- 函数 `matGetVariable()`: 从 MAT 文件中读取 MATLAB 数据。
- 函数 `matPutVariable()`: 向 MATLAB 的 MAT 文件中写入数据。
- 函数 `matGetNextVariable()`: 从 MAT 文件中读取下一个 MATLAB 数据。
- 函数 `matDeleteVariable()`: 从 MAT 文件中删除 MATLAB 数据。
- 函数 `matPutVariableAsGlobal()`: 向 MATLAB 的 MAT 文件中写入数据，并设置其为全局变量。
- 函数 `matGetVariableInfo()`: 从 MAT 文件中读取 MATLAB 数据的头信息。
- 函数 `matGetNextVariableInfo()`: 从 MAT 文件中读取下一个 MATLAB 数据的头信息。

【例 9.3】MAT 文件的使用。

```
>> clear                                %清除当前工作空间内的变量
>> a1=5;                                %变量的定义
>> a2=rand(2);
>> b1='s';
>> b2={0,1,2};
>> save mattest;                         %以文件名为 mattest 保存变量
>> clear
>> load mattest a*;                      %向 MATLAB 中导入数据，且只导入以字符“a”起始的变量
```

9.4 MATLAB 引擎技术

MATLAB 引擎技术是为用户提供了一种可直接在其他程序中调用 MATLAB 函数的技术,通过该技术的使用可以方便地调用 MATLAB 的各种函数,即前台程序调用 MATLAB 函数,后台 MATLAB 根据前台的调用执行相关的命令、操作。MATLAB 引擎技术简化了前台的工作量,但是该技术本质还是需要在后台运行 MATLAB 的程序,因而程序无法脱离 MATLAB 执行。

MATLAB 引擎技术提供了引擎函数应用在其他程序语言中控制 MATLAB 中的操作。以 C 语言为例,总共提供了 13 个引擎函数,以字符“eng”为前缀,包含于头文件 engine.h 中。下面简要介绍引擎库中的几个常用的引擎函数。

- 函数 engOpen(): 用于启动 MATLAB 引擎,函数的定义为 `Engine *engOpen(const char *startcmd)`, 其中输入参数 startcmd 为字符串,用于启动 MATLAB 进程。
- 函数 engClose(): 用于关闭 MATLAB 引擎,函数的定义为 `int engClose(Engine *ep)`, 其中输入参数 ep 为需要关闭的引擎的指针。
- 函数 engEvalString(): 用于执行 MATLAB 的一个字符串的表达式,函数的定义为 `int engEvalString(Engine *ep, const char *string)`, 其中参数 string 为 MATLAB 的字符串表达式。
- 函数 engGetArray(): 用于复制 MATLAB 工作空间中变量,其函数定义为 `mxArray *engGetArray(Engine *ep, const char *name)`, 其中 name 为复制的变量名。
- 函数 engPutArray(): 用于将变量写入 MATLAB 工作空间中,其函数的定义为 `int engPutArray(engine *ep, const mxArray *mp)`, 其中参数 mp 为写入的变量名。

在 C 或 C++ 等高级语言中使用 MATLAB 引擎的步骤为: 首先,必须先要打开一个引擎,并获取该引擎的指针,调用时通过引擎函数库中函数来向 MATLAB 发送需要执行的命令,然后再通过引擎函数将执行的程序返回应用程序中。引擎函数在 C 或 C++ 等高级语言调用 MATLAB 命令中发挥了重要的作用,同时在定义程序头文件时必须定义引擎函数。

在使用 MATLAB 引擎功能时,对高级语言也需要进行一些设置,如果为 VC 编译环境,则需要设置以下几步。

(1) 添加引擎的头文件 engine.h 所在目录到 include files: 选择 VC 菜单中的“Tool”→“Options”命令,在打开的“Options”对话框的“Directories”选项卡中,在“Include file”中添加 MATLAB\extern\include,如图 9.2 所示。

(2) 添加引擎用到的动态连接库的目录: 选择 VC 菜单中的“Tool”→“Options”,在打开的“Options”对话框的“Directories”选项卡中,在“Library files”中添加 MATLAB\extern\include,如图 9.3 所示。

(3) 添加引擎的 3 个库文件: 在 VC 中,选择“Project”→“Setting”命令,在打开的“Project setting”对话框中的“Link”选项卡中,在“Object/library modules”中添加“libmx.lib”、“libmat.lib”、“libeng.lib”,如图 9.4 所示。

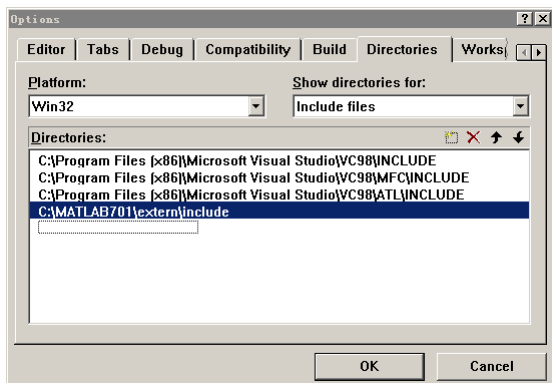


图 9.2 添加引擎的头文件 engine.h

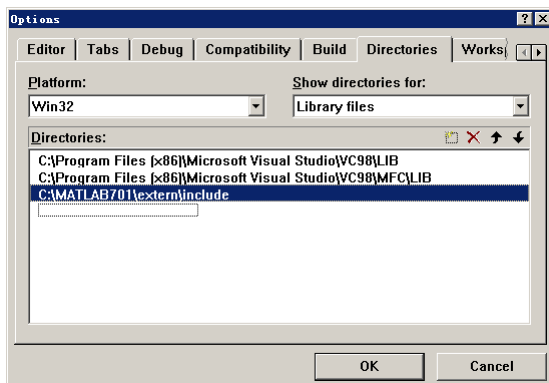


图 9.3 添加引擎的动态连接库

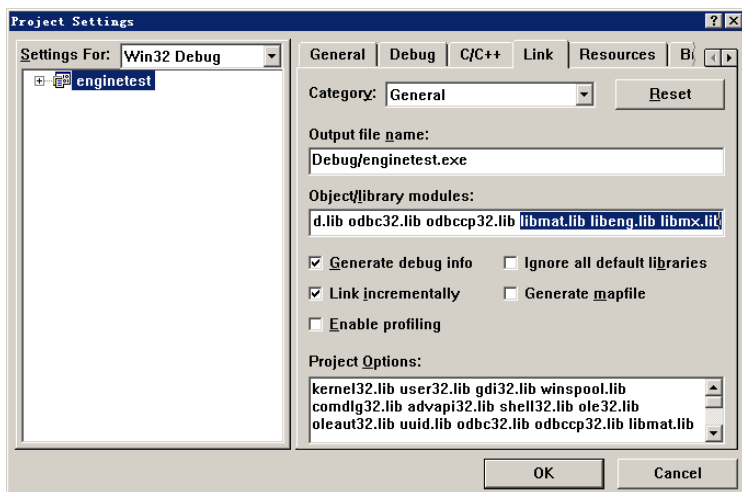


图 9.4 添加引擎的 3 个库文件

【例 9.4】MATLAB 引擎技术的使用。

```
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <engine.h>                                     //说明 MATLAB 引擎的头文件

int main()
{
    Engine *ep;                                         //定义 MATLAB 引擎的指针
    mxArray *t=NULL;                                   //函数调用的输入参数定义
    mxArray *result=NULL;                              //函数调用的输出参数定义
    int x[10]={2,4,6,8,10,12,14,16,18,20};
    ep=engOpen(NULL);                                  //打开 MATLAB 引擎
    t=mxCreateDoubleMatrix(1,3,mxREAL);               //创建数组
    memcpy(mxGetPr(t),x,sizeof(x));                   //复制数据到数组中
    engPutVariable(ep,"t",t);                          //将变量 t 写入 MATLAB 中
    engEvalString(ep,"y=sin(t/10)");                  //执行 MATLAB 字符串命令
    engEvalString(ep,"plot(y)");                      //执行 MATLAB 字符串命令
}
```

执行上述的 C 程序，将在后台调用 MATLAB 程序，生成如图 9.5 所示的结果。

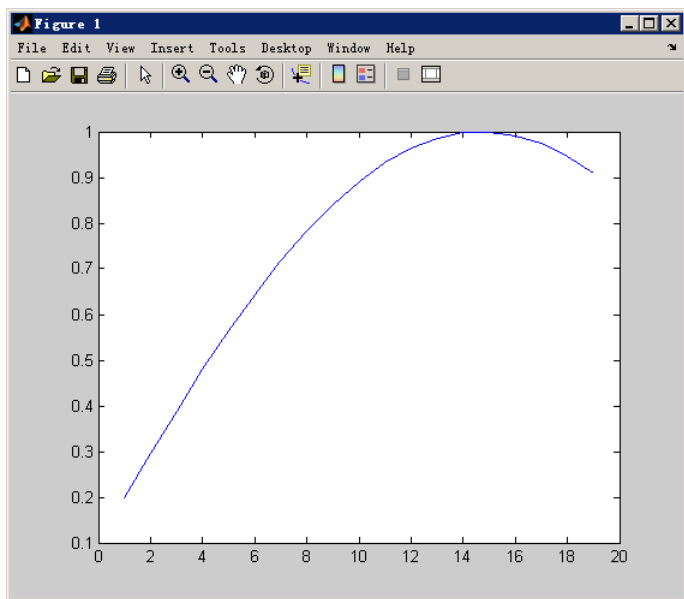


图 9.5 MATLAB 引擎技术的使用

9.5 COM 组件

COM (Component Object Model, 组件对象模型) 是以组件为发布单元的对象模型, 是面向对象的技术。COM 组件是建立在二进制级别上的规范, 所以组件的接口编程不受程序语言类型的限制。MATLAB 中的 COM 生成器能把 MATLAB 开发的算法做成独立的组件, 这些组件可以直接被 C、C++、VB、C#、JAVA 或其他支持 COM 的语言所使用。

COM 组件是已编译好的代码, 因而执行效率高, 可弥补 MATLAB 作为解释性的语言在代码执行效率上不高的缺陷。同时, 使用组件不需要重新编译整个程序, 在不修改程序的情况下可以重新发布组件。

COM 组件技术是利用 MATLAB 中的 MATLAB COM Builder 工具, 将 M 文件转换成 dll 文件, 然后在其他编程语言中调用该 dll 文件。下面以一个实例演示如何利用 MATLAB 的 COM 生成器创建可供其他程序调用的 dll 文件。

【例 9.5】MATLAB COM 组件的使用。

基于 MATLAB 生成组件的步骤为:

(1) 根据功能要求编写 MATLAB 的 M 函数文件。

```
function y=comstudytest(x)
y=rand(x);
end
```

(2) 在命令窗口输入 “comtool” 命令, 进入如图 9.6 所示的 “MATLAB Builder” 窗口, 选择 “File” → “New Project” 命令打开 “New Project Settings” 窗口, 新建工程并设置相关的参数, 如图 9.7 所示, 设置创建的组件的名称为 “comtest”、组件的类为 “comtestclass” 及生成的组件在磁盘上的保存位置。

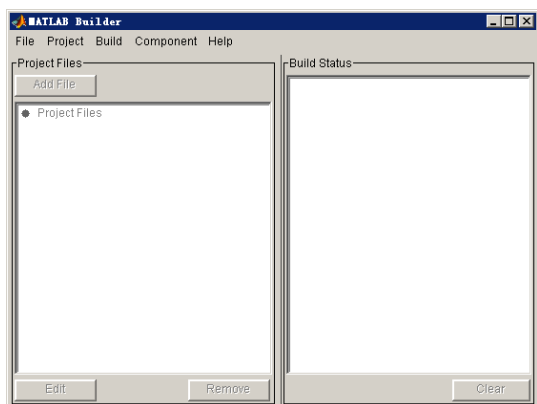


图 9.6 “MATLAB Builder” 窗口

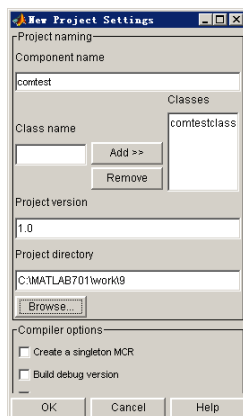


图 9.7 “New Project Settings” 窗口

(3) 返回“MATLAB Builder”窗口,选中“M-files”,单击“Add File”按钮,在新创建的类中添加 M 文件,把步骤(1)中编写的 M 文件 comstudytest.m 添加到 comtestclass 类的 M-files 中,如图 9.8 所示。

(4) 选择“Build”→“Com Object”命令,创建 M 文件的 COM 对象,如图 9.9 所示。

(5) 最后,对生成的 COM 组件进行打包,选择“Component”→“Package Component”命令,打开如图 9.10 所示的“Package Files”窗口,单击窗口中的“Create”按钮,实现组件的打包。

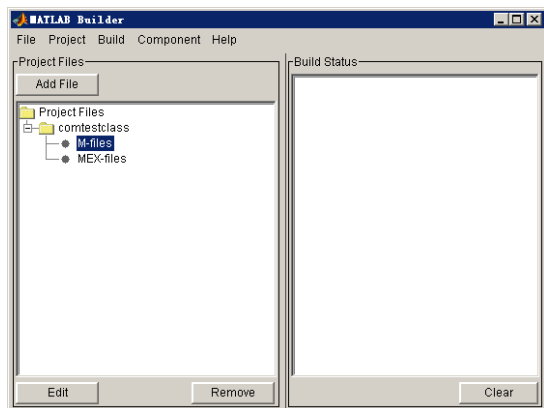


图 9.8 在新建的类中添加 M 文件

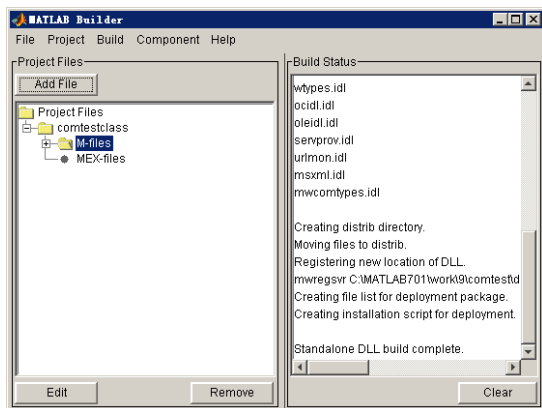


图 9.9 COM 组件的生成

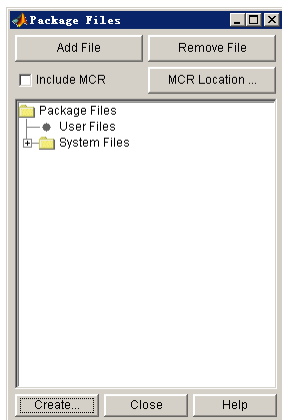


图 9.10 “Package Files” 窗口

至此，通过 MATLAB 的 COM 生成器，已完成 M 文件到 COM 组件的编译工作，在路径下生成“comtest_1_0.dll”文件，该文件可供不同的应用程序调用。关于在不同程序中调用，比较简单，只要向组件提供输入参数，调用组件，即可获得输出参数。不同的程序语言中可能有所差异，读者可以在实际使用时再查阅相关的技术文档。

基于 COM 组件的混合编程技术生成的程序可脱离 MATLAB 直接运行，在程序发布到其他人的电脑上时无须安装 MATLAB 软件，但是需要安装 MCR。MCR 为保证 M 文件可独立执行的函数库，一般在发布基于 MATLAB COM 对象开发的应用程序时，需要同时发布 MCR 的程序。MCR 程序无须重新下载，在用于开发的 MATLAB 版本的安装目录下。笔者将 MATLAB 安装在 C 盘，即 MCR 程序所在的路径为“C:\MATLAB701\toolbox\compiler\deploy\win32”，在其中读者可以看到一个文件名为 MCRIInstaller 的应用程序，将其复制出来与设计好的应用程序同时发布。

用户在使用基于 COM 组件设计的程序的时候，需要首先安装 MCR 程序，并进行相关的注册工作，下面简单介绍一下 MCR 的安装工作。

【例 9.6】MCR 的安装。

(1) 双击“MCRIInstaller.exe”，进入如图 9.11 所示的安装界面。

(2) 单击两次“Next”按钮，进入如图 9.12 所示的选择安装路径的界面，一般采用默认安装设置。

(3) 再次单击“Next”按钮，开始进行如图 9.13 所示的 MCR 安装。

(4) 安装完毕后，将显示如图 9.14 所示的窗口。

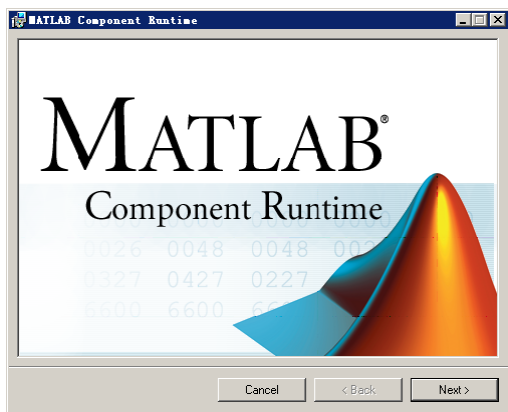


图 9.11 MCR 的初始安装界面

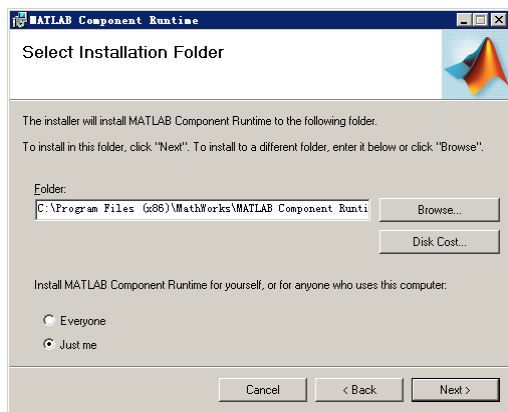


图 9.12 MCR 路径的选择

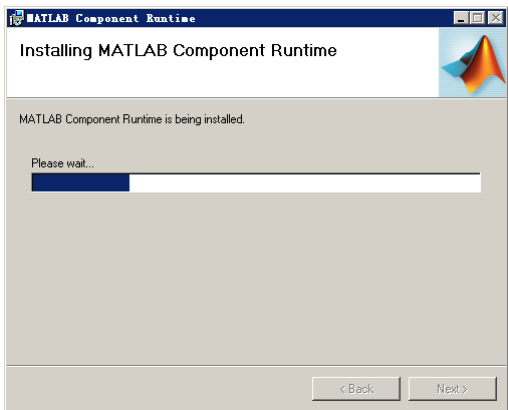


图 9.13 MCR 的安装

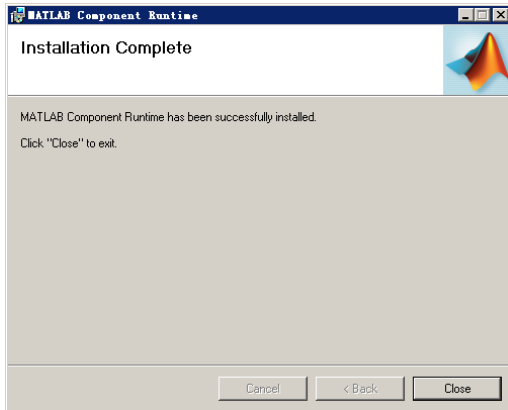


图 9.14 MCR 安装完毕窗口

9.6 与 Word、Excel 的混合使用

Word 与 Excel 是日常工作中最为常用的办公软件，而 MATLAB 的数据分析结果也往往需要整理到 Word 或 Excel 中。MATLAB 提供了对于 Excel 和 Word 的支持，使我們可以在 Excel 和 Word 中直接调用 MATLAB 程序，并将结果返回到 Word 和 Excel。

通过 MATLAB 与 Word 和 Excel 的混合使用，我们可以充分利用 Word 强大的文字编辑处理功能，Excel 简单直观的数据显示、做图、分析功能和 MATLAB 专业的数值计算、图形处理、模拟仿真等功能。本节将向读者介绍如何在 Word 和 Excel 中使用 MATLAB，为我们的工作带来更大的便利。

9.6.1 Excel Link 的使用

在 Excel 中使用 MATLAB 需要通过插件 Excel Link，该插件用于在 Windows 环境下实现 Excel 与 MATLAB 的无缝链接。通过 Excel Link 用户可以在 Excel 工作表空间和宏编程工具中调用 MATLAB 的各项功能。

1. Excel Link 的安装和配置

Excel Link 插件在使用前需要先安装，插件的安装工作在 MATLAB 软件安装中进行。如果用户选择了典型安装，其默认状态下 Excel Link 已安装好；如果选择的安装方式为自定义，则需要安装 MATLAB 软件的时候选择“Excel Link”复选框，如图 9.15 所示。

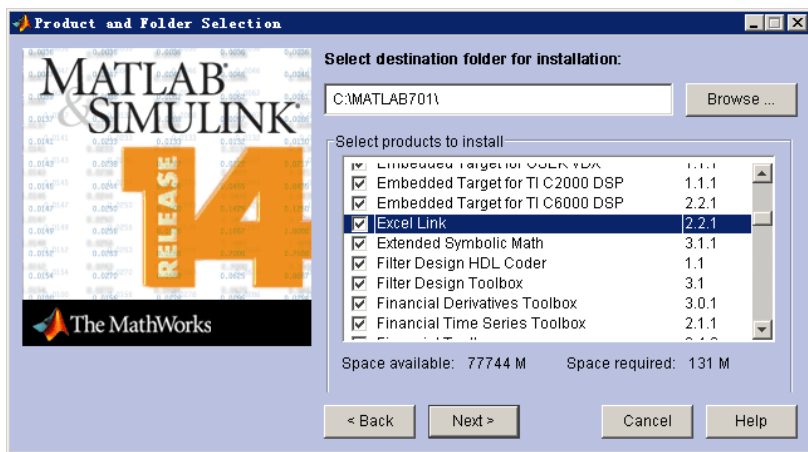


图 9.15 Excel Link 的安装

在安装好 Excel Link 插件后，在 Excel 中使用时还需要进行相关的一些配置。启动 Excel 2003，选择“工具”→“加载宏”命令，弹出如图 9.16 所示的对话框。选择“Excel Link 2.2.1 for use with MATLAB”复选框，如果不存在则浏览目录，在 MATLAB 安装目录下的 toolboxexlink 文件夹中找到 exclink.xla 文件，加载此文件。

在 Excel 中菜单栏的左侧多了一个 Excel Link 工具条，如图 9.17 所示。经过以上的设置后，Excel Link 可以正式开始使用了，在以后的使用中无须重新配置。

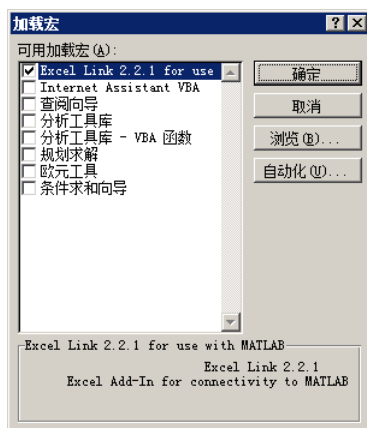


图 9.16 Excel Link 的配置

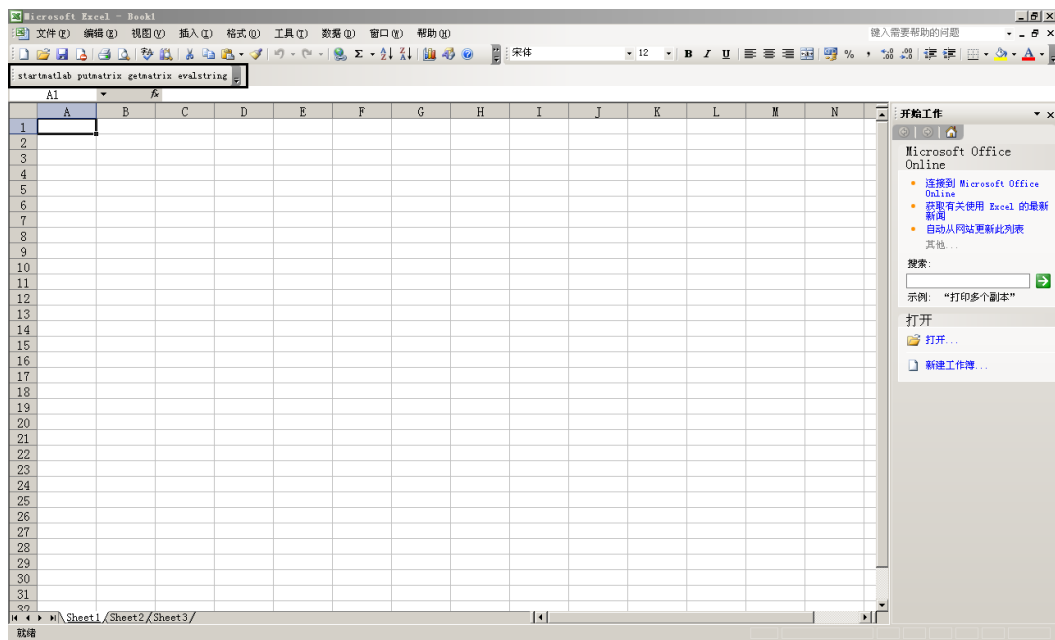


图 9.17 配置好 Excel Link 的 Excel 界面

2. Excel Link 的使用

Excel Link 的使用是在 Excel 环境下完成的, 所以很多操作方式与 Excel 更为接近。Excel Link 功能的实现主要通过 13 个函数, 其中 4 个为链接管理函数, 9 个为数据管理函数, 函数的调用方式与 Excel 函数的使用一样, 单元格的调用方式即在 Excel 工作表的单元格中输入 “=function_name(variable)” 即可, 需要注意的是括号内的变量需加双引号; 宏命令的调用方式即直接把函数作为宏命令执行。下面简单介绍一下这 11 个函数。

1) 链接管理函数

链接管理函数主要用于控制 Excel Link 与 MATLAB 的链接和 MATLAB 进程的启动与关闭, 4 个链接管理函数如下。

- 函数 `MLAutoStart()`: 设置是否自动启动 MATLAB, 参数可选值为 “no” 或者 “yes”, 如果使用参数 “yes”, 则当 Excel 启动时, 自动启动 MATLAB 和 Excel Link, 如果使用参数 “no”, 则当 Excel 启动时, 不启动 MATLAB 和 Excel Link。

- 函数 MATLABinit(): 用于初始化 Excel Link 和启动 MATLAB, 该函数只能以宏命令的方式运行, 而其他的 3 个数据链接函数可以在数据单元格中使用或者按照宏命令执行。
- 函数 MLClose(): 用于关闭 MATLAB 进程并删除 MATLAB 工作空间的所有变量。
- 函数 MLOpen(): 用于启动 MATLAB 进程。

2) 数据管理函数

Excel Link 提供了 9 个数据管理函数, 实现 MATLAB 与 Excel 之间数据的传递, 同时可在 Excel 中执行 MATLAB 命令。其中:

- 函数 MATLABfcfn(): 根据指导的 Excel 数据执行 MATLAB 命令, 函数的调用格式为 matlabfcfn(command,inputs), 其中输入参数 command 为需要 MATLAB 执行的命令, 参数 inputs 用于确定执行 MATLAB 语句的数据, 为 Excel 中的包含数据的工作表单元格范围, MATLAB 执行后返回的结果将传递到调用该函数的单元格中。
- 函数 MATLABsub(): 用于使用指定的 Excel 数据运行 MATLAB 命令, 并在指定的 Excel 单元格位置输出计算结果。该函数的调用格式为 matlabsub(command,edat,inputs), 其中参数 command 为 MATLAB 的执行命令, edat 为调用 MATLAB 计算后返回的结果存储的 Excel 单元格位置, input 参数为执行命令的输入数据在 Excel 中的单元格的位置。
- 函数 MLDeleteMatrix(): 用于删除 MATLAB 工作空间内的变量, 其调用格式为 MLDeleteMatrix(var_name), 其中输入参数 var_name 为要删除的 MATLAB 工作空间内的变量名。
- 函数 MLEvalString(): 执行 MATLAB 的字符串命令, 函数的调用格式为 MLEvalString(command), 其中输入参数 command 为需要执行的命令语句。
- 函数 MLGetMatrix(): 用于在 Excel 数据表中写入 MATLAB 中的数据, 其调用格式为 MLGetMatrix(var_name,edat), 其中函数输入参数 var_name 为在 Excel 中写入的 MATLAB 变量名, edat 为写入变量在 Excel 中的单元格位置。
- 函数 MLGetVar(): 将 MatLab 工作空间内的数据传送给 Excel VBA 变量, 只能在宏命令中使用, 其调用格式为 MLGetVar(ML_var_name, VBA_var_name), 其中参数 ML_var_name 是 MATLAB 中即将获取的数据名, 参数 VBA_var_name 为 MATLABB 数据传给 VBA 变量名。
- 函数 MLAppendMatrix(): 用于向 MATLAB 工作空间内的变量后添加 Excel 中的数据, 函数的调用格式为 MLAppendMatrix(var_name,mdat), 其中输入参数 var_name 为 MATLAB 工作空间内需要添加数据的变量, 参数 mdat 为添加的数据在 Excel 中的单元格位置, 同时需要注意添加的数据维数要和原矩阵中的维数相匹配。
- 函数 MLPutMatrix(): 使用指定的 Excel 工作表中的数据, 创建或者覆盖 MATLAB 变量, 其调用格式为 MLPutMatrix(var_name,mdat), 其中参数 var_name 为待创建或者覆盖的 MATLAB 变量, mdat 为数据在 Excel 中的单元格位置。
- 函数 MLPutVar(): 使用 VBA 变量创建或者覆盖 MATLAB 数据, 该函数只能在宏程序中使用, 其调用的语法为 MLPutVar (ML_var_name, VBA_var_name), 其中, ML_var_name 为 MATLAB 中的变量名, VBA_var_name 为 VBA 变量名。

【例 9.7】Excel Link 的使用。

现以一班级部分同学的成绩数据为例, 演示 Excel Link 的使用, 四列数据分别为学号、数学成绩、语文成绩和英语成绩。

(1) 调用 MATLAB 的均值函数计算各门课程的平均成绩, 如图 9.18 所示为直接使用 MATLAB 的函数。在输出均值成绩的 Excel 单元格中输入 “=matlabfcfn("mean","B2:B21)", 其中 mean 代表

需要调用的 MATLAB 的函数名, B2:B21 为计算均值的数据的单元格标识。按回车键, MATLAB 即可计算并返回结果, 类似的可以计算出语文、英语的平均成绩。

(2) 在 Excel 中的任意单元格中输入 “=MLPutMatrix(grade,B2:D21)”, 在 MATLAB 工作空间内创建成绩变量 grade。

(3) 通过 Excel Link 在 MATLAB 中计算各门成绩的标准差并返回到 Excel 中的相应位置, 在 Excel 中的任意单元格中输入 “=MLEvalString('y=std(grade)')”, 计算成绩的标准差并返回到 MATLAB 中的变量 y 中; 然后再在 Excel 中的任意单元格中输入 “=MLGetMatrix('y','sheet1!B23')”, 从 MATLAB 中获取计算的值, 并返回到 Excel 中相应的位置, 如图 9.19 所示。

	A	B	C	D	E
1	学号	数学	语文	英语	
2	101	98	75	80	
3	102	67	86	87	
4	103	88	87	85	
5	104	78	82	84	
6	105	79	73	78	
7	106	98	86	76	
8	107	95	90	79	
9	108	90	78	90	
10	109	87	86	68	
11	110	88	86	90	
12	111	70	76	86	
13	112	85	79	88	
14	113	75	69	76	
15	114	95	88	77	
16	115	94	90	74	
17	116	87	82	73	
18	117	82	78	80	
19	118	76	74	80	
20	119	45	84	88	
21	120	78	67	89	
22		=matlabfcn("mean",B2:B21)			
23					

	A	B	C	D	E
1	学号	数学	语文	英语	
2	101	98	75	80	
3	102	67	86	87	
4	103	88	87	85	
5	104	78	82	84	
6	105	79	73	78	
7	106	98	86	76	
8	107	95	90	79	
9	108	90	78	90	
10	109	87	86	68	
11	110	88	86	90	
12	111	70	76	86	
13	112	85	79	88	
14	113	75	69	76	
15	114	95	88	77	
16	115	94	90	74	
17	116	87	82	73	
18	117	82	78	80	
19	118	76	74	80	
20	119	45	84	88	
21	120	78	67	89	
22	平均成绩	82.75	80.8	81.4	
23	标准差	12.65275	6.856422	6.369334	
24					

图 9.18 Excel Link 的使用 (MATLAB 函数使用) 图 9.19 Excel Link 的使用 (MATLAB 与 Excel 的数据传递)

(4) 在 Excel 中调用 MATLAB 函数绘制柱形图。在 Excel 中的任意单元格输入函数 “=MLEvalString('bar(grade)')” 将绘制出如图 9.20 所示的显示成绩的柱形图。

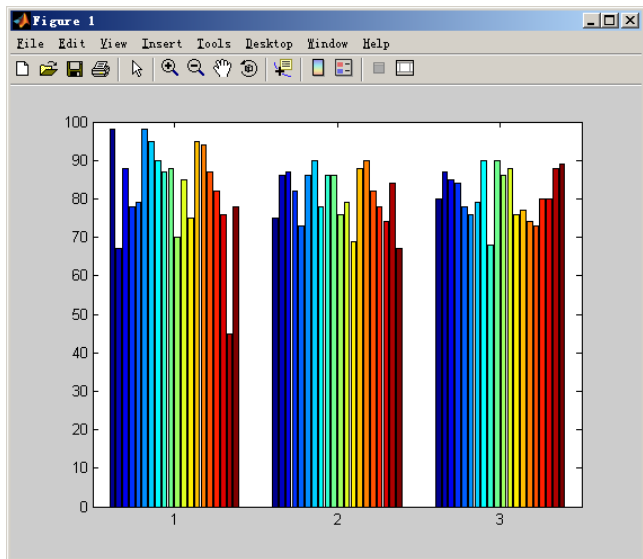


图 9.20 Excel Link 的使用 (绘制柱形图)

(5) 清空 MATLAB 内存空间的变量, 并安全退出 MATLAB 软件。在 Excel 中的任意单元格中输入命令 “MLClose()” 将退出 MATLAB, 并清空 MATLAB 中的数据。

9.6.2 在 Word 中使用 Notebook

Notebook 是 Math Work 将 Word 与 MATLAB 集成为一体,使用户在文字处理的同时可以方便地进行 MATLAB 的图形处理、工程计算等。Notebook 工具的核心是 M-book 模板, M-book 模板文档同时具有 Word 和 MATLAB 的所有功能,用户在该文档下可以方便地进行文字处理的相关工作,同时后台可以方便地调用 MATLAB 执行操作,计算结果可直接返回到 Word 中。

本小节中将向读者详细介绍如何在 Word 中使用 Notebook,包括 Notebook 的安装、Notebook 的启动。

1. Notebook 的安装

启动 MATLAB,在其命令窗口输入如下命令。

```
>> notebook -setup
%选择需要安装 Notebook 的 MATLAB 版本
Welcome to the utility for setting up the MATLAB Notebook
for interfacing MATLAB to Microsoft Word
Choose your version of Microsoft Word:
[1] Microsoft Word 97
[2] Microsoft Word 2000
[3] Microsoft Word 2002 (XP)
[4] Microsoft Word 2003 (XP)
[5] Exit, making no changes
Microsoft Word Version: 4
```

在此,我们选择 Word 2003 (XP),此时 MATLAB 会自动寻找 Word 的安装路径,并在路径下寻找模板文件 normal.dot。如果找到了,则出现以下提示信息。

```
Notebook setup is complete.
```

“Notebook setup is complete”表示 Notebook 安装结束。

2. Notebook 的启动

Notebook 的启动方式主要有两种,分别是 MATLAB 或者 Word 中启动,下面详细介绍这两种启动方式,读者可以根据个人喜好选择启动方式。

1) 从 MATLAB 中启动 Notebook

启动 MATLAB,在命令窗口输入命令 Notebook,将打开以 M-book 为模板的 Word 文档,如图 9.21 所示,与一般的 Word 文档不同的是在菜单栏中的表格菜单项的右边增加了一个 Notebook 菜单项。

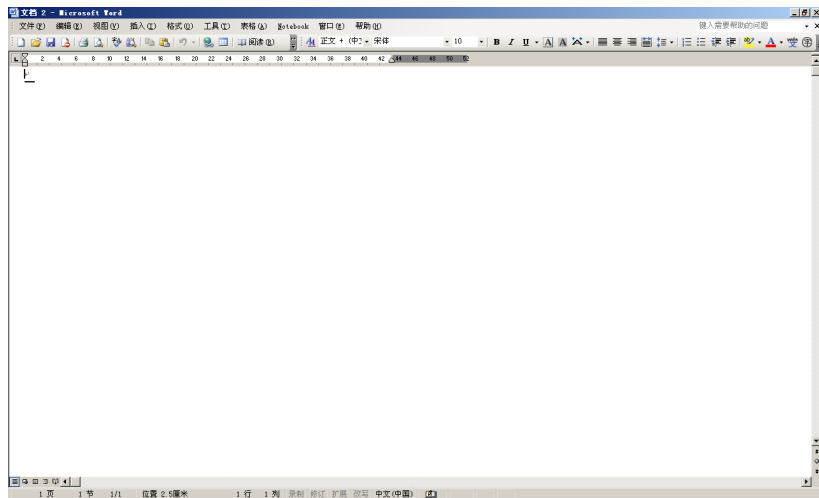


图 9.21 Notebook 的使用

2) 从 Word 中启动 Notebook

启动 Word 程序，然后选择“文件”→“新建”命令，选择“本机上的模板”选项，打开如图 9.22 所示的“模板”对话框。在“常用”选项卡中选择“m-book.dot”模板，单击“确定”按钮即可启动 Notebook。



图 9.22 从 Word 中启动 Notebook

3. Notebook 的运行机制

Notebook 是通过动态链接实现与 MATLAB 的交互，其交互链接的基本单位为 cell (细胞)。Notebook 将需要 MATLAB 执行的命令组成 cell，再传送到 MATLAB 中运行，运行结果再以 cell 的形式返回 Notebook。输入 MATLAB 中的细胞可以是一行或多行文本命令，而输出的细胞也可以是一个或多个变量。

4. Notebook 菜单

Notebook 的运行机制主要依靠其基本单元细胞，而细胞的创建、执行等主要通过 Notebook 菜单来实现，如图 9-23 所示，下面具体讲述 Notebook 菜单各菜单项的功能，最后以一个实例演示如何使用 Notebook。

Notebook 菜单的菜单项如下。

- Define Input Cell: 定义输入细胞，选择 Word 中需要 MATLAB 执行的代码，单击该菜单项或按“Alt+D”组合键，执行命令文本变为绿色，表示该部分文本为 Notebook 的输入细胞。
- Define AutoInit Cell: 定义自动初始化细胞，类似于输入细胞的定义，选中需要定义的文本，单击该菜单项，文本变为深蓝色，自动初始化细胞是在程序初始打开时就自动执行的代码。
- Define Calc Zone: 定义计算区，一般的 Notebook 由文字描述、输入细胞、输出细胞等多部分组成，而计算区的使用可以方便管理文件，我们可以把所有的输入细胞内容定义为计算区，用户选择执行计算区即可一次执行计算区内的所有代码。
- Undefine Cells: 取消细胞的定义，将细胞转换为文本，使用方法同其他菜单。
- Purge Selected Output Cells: 删除所选区域内的所有输出细胞，可以重新输出结果。
- Group Cells: 定义细胞组，当用户需要把若干条 MATLAB 命令组合在一起时，可以同时选中输入细胞或者自动初始化细胞，然后单击此菜单，选中的细胞即建立了细胞组。
- Ungroup Cells: 取消细胞组的定义，细胞组转换为输入细胞或者自活细胞。
- Hide(Show) Cells Markers: 隐藏(显示)细胞中的标识符中括号。
- Toggle Graph Output for Cell: 输出细胞内是否嵌入生成的图形。

- Evaluate M-book: 执行 Notebook 中的所有输入细胞。
- Evaluate Loop: 循环运行输入细胞, 选中需要循环运行的输入细胞, 单击此菜单项, 打开如图 9.24 所示的对话框, 在其中设置循环运行的参数, 其中“Stop After”设置循环终止的次数, “Loop Speed”中的“Slower”按钮表示循环后延迟, “Start”按钮用于启动循环运行细胞, “Pause”按钮用于暂停循环运行的细胞, “Close”按钮用于关闭循环运行。
- Evaluate Cell: 用于运行输入细胞, 单击该菜单或按“Ctrl+Enter”组合键就可以把定义的输入细胞传递到 MATLAB 中执行, 执行结果将会自动返回到 Word 中, 运算结果标识为蓝色。
- Evaluate Calc Zone: 用于执行计算区的细胞内容。
- Bring MATLAB to Front: 用于将 MATLAB 命令窗口调到前台。
- Notebook Options: 设置 Notebook 使用的参数, 单击该菜单项将打开如图 9.25 所示的对话框, 其中 Numeric Format 区域用于设置输出结果的显示数据类型和数据之间的间隔(松或紧); Figure Options 区域用于设置输出结果中嵌入 Word 中图形的大小。

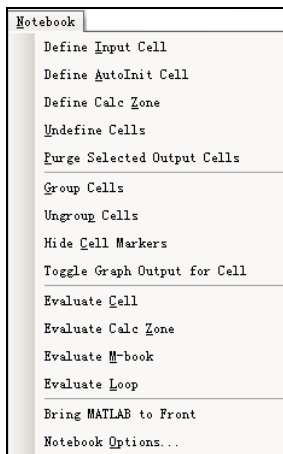


图 9.23 Notebook 菜单

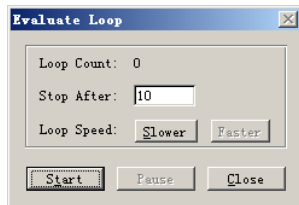


图 9.24 “Evaluate Loop”对话框

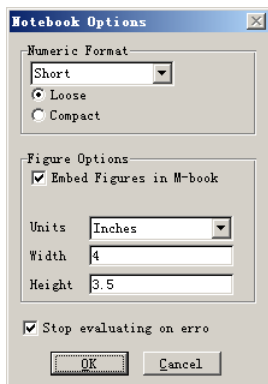


图 9.25 “Notebook Options”对话框

【例 9.8】Notebook 的使用。

打开 Notebook 文档, 在其中输入以下文本。

```

Notebook 程序演示
x=0:pi/100:pi;
plot(sin(x));
a=magic(3)

```

选中下面的 3 行命令，选择“Notebook”→“Define Input Cell”命令，输入的文本将变为绿色文本。继续选择“Notebook”→“Evaluate Cell”命令，运行输入细胞，后台将调用 MATLAB 的程序，同时在 Word 中返回执行的结果，如图 9.26 所示。

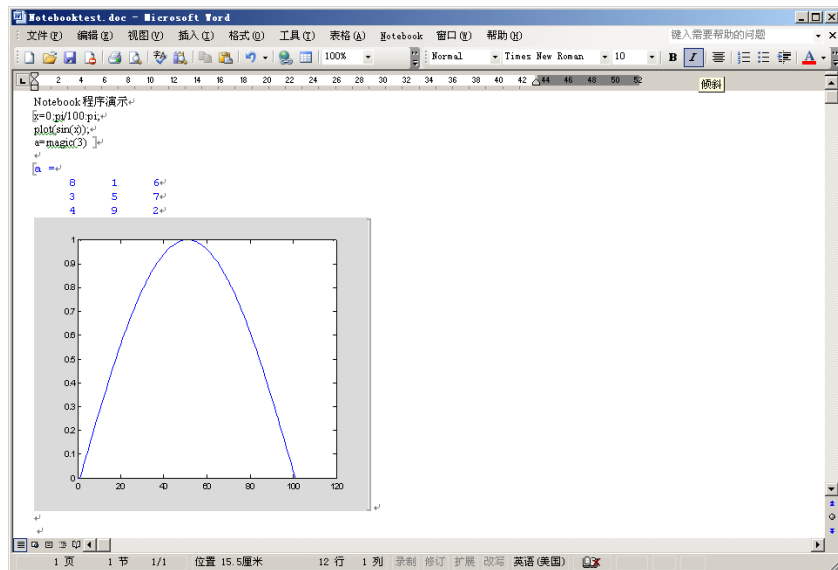


图 9.26 Notebook 的使用

9.7 本章小结

本章主要向读者简单介绍了 MATLAB 接口编程的相关知识。通过接口编程技术，用户可以在 MATLAB 中调用其他高级语言的程序，同时也可在其他高级语言中调用 MATLAB 的程序。在实际应用中读者可根据实际的需要，充分利用各编程语言的优势，使用接口编程技术，简化编程、调高编程效率。本章限于篇幅，对各接口编程技术仅做了简单的介绍，读者通过书中的介绍可以确定需要采用的接口编程技术，如果想进一步学习可以查阅相关文献资料。

在本章的最后一节，向用户介绍了如何在最常用的两个 Office 软件中使用 MATLAB，这部分内容较为实用，感兴趣的读者在平时就可以尝试使用，相信会给学习、工作带来不少便利和乐趣。



第10章

文件 I/O

在前面的章节中读者已经学习了 MATLAB 常用的基本操作，但在实际使用过程中读者首先会遇到如何科学合理地导入需要分析的数据。在前面章节的实例中由于仅为演示 MATLAB 函数的使用，函数的输入数据多是通过命令行直接生成，而在实际中我们要分析的数据往往很复杂，同时数据在磁盘多是以非 MATLAB 文件形式存在，因而在处理复杂问题时，用户首先需要面对的是如何把数据准确地导入 MATLAB 工作空间中进行运算。

同时，利用 MATLAB 分析完数据后，把结果有效地保存下来也是很有必要的。MATLAB 在退出后工作空间内的数据会自动清空，因而实时地保存重要的结果是必须的。另外，我们也更希望结果能保存在一些易读、易修改的 Excel 或 txt 文本文件中。

本章将详细地介绍 MATLAB 文件的输入/输出机制。通过本章的学习，读者将了解数据文件、图片文件在 MATLAB 中的导入和导出。

10.1 数据文件

数据文件的 I/O 可以利用菜单中的选项在界面中直接导入/导出数据文件，或者通过函数法实现。其中函数操作又可根据函数的类型分为低级文件的 I/O 操作和高级文件的 I/O 操作。低级文件的 I/O 操作读/写数据比较复杂，需要设置较多的参数，但是相对比较灵活，而高级文件的 I/O 操作是 MATLAB 向用户提供的可较为简洁地导入/导出数据的函数。

10.1.1 低级文件的 I/O 操作

低级文件的 I/O 操作大致的操作过程为打开文件并标识打开的文件，输入从文件中读入或写入，完成操作后关闭文件。下面具体介绍实现这些功能的函数及其使用方法。

1. 文件的打开

函数 `fopen()` 用于文件的打开，并需指定文件打开后进行操作的方式。对于存在的文件会自动在当前路径下创建文件并打开。同时，数据文件的格式主要有二进制文件和文本文件，在打开文件时需要指定打开文件的类型。对于打开的文件还需要获取文件句柄值用于下一步文件的具体操作，同时句柄值也可判断文件打开是否成功。

函数 `fopen()` 的调用格式如下。

- `fid = fopen(filename)`: 打开文件名为 `filename` 的文件，并返回文件句柄 `fid`，如果返回的句柄值为 -1，表示文件打开不成功，返回的句柄值大于 0，则表示打开文件成功，`fid=1` 表示标准输出文件，`fid=2` 表示标准误差文件，用户打开的文件句柄值从 3 开始标识，文件关闭后，句柄值将失效，默认状态下以只读方式打开文件，并且打开方式默认值为二进制文件。
- `fid = fopen(filename, mode)`: 打开文件名为 `filename` 的文件，并设置其打开模式 `mode`，

其中 r (以只读方式打开文件, 文件必须已存在), r+ (以读/写方式打开文件, 打开后先读后写, 文件必须已存在), w (以写方式打开文件, 如果文件已存在则等待执行写操作, 如果文件不存在, 则创建新的文件), w+ (以读/写方式打开文件, 先读后写, 如果文件已存在则等待执行读/写操作, 如果文件不存在, 则创建新的文件), a (以写方式打开文件, 写入位置为文件末尾, 如果文件已存在则等待执行写操作, 如果文件不存在, 则创建新的文件), a+ (以读/写方式打开文件, 先读后写, 写入位置为文件末尾, 如果文件已存在则等待执行写操作, 如果文件不存在, 则创建新的文件), 另外, 在以上的打开模式字符串后加字符 “t” 表示以文本方式打开, 添加字符 “b” 表示以二进制方式打开, 默认是以二进制方式打开, 文本文件为只存储文本字符数据的文件, 其他的文件就是二进制文件。

- [fid,message] = fopen(filename, mode, machineformat): 打开文件名为 filename 的文件, 参数 machineformat 用于设置读/写时数据格式, 默认为 ieee-le。
- fids = fopen('all'): 返回当前所有打开的文件的句柄值。
- [filename, mode, machineformat] = fopen(fid): 返回被打开句柄为 fid 的文件信息, 包括文件名 filename, 文件的打开方式 mode, 文件读/写时的数据格式 machineformat。

【例 10.1】文件的打开

```
>> fid=fopen('fopentest.txt')           %以只读方式打开文件, 文件不存在, 返回“-1”的句柄值
fid =
    -1
>> fid=fopen('fopentest.txt','w')       %以写方式打开文件, 文件不存在, 创建文件, 并返回句柄值 3
fid =
     3
>> fid=fopen('fopentest2.txt','ab')     %以写方式打开文件, 文件不存在, 创建文件, 并返回句柄值 4
fid =
     4
>> fids = fopen('all')                  %返回当前打开的所有文件的句柄值
fids =
     3     4
>> [filename, mode, machineformat] = fopen(3) %返回句柄值为 3 的文件信息
filename =
fopentest.txt
mode =
wb
machineformat =
ieee-le
fclose(fids)
```

使用函数 fopen() 打开文件的一些注意事项:

(1) 根据需要对文件进行的读/写操作设置文件的打开模式, 可以避免不必要的读/写错误, 例如仅需要对文件进行读的操作, 以只读方式打开是最好的, 可以避免对文件错误的写入, 覆盖原来的内容。

(2) 文件打开后及时地检查文件是否打开成功, 给用户以反馈, 打开失败, 后面的程序将无法正常运行。

(3) 文件打开后要及时地关闭。

2. 文件的写入

MATLAB 采用了不同的函数用于二进制文件和文本文件的写入操作, 其中, 函数 fwrite() 用于二进制文件的写入操作, 其调用格式如下。

- count = fwrite(fid,A,precision): 数据 A 将以精度 precision 写入句柄值为 fid 的已打开的文件, 返回 count 值, 写入成功, 则返回写入数据的个数, 否则返回 0, 其中常用的数据精度包括 char (字符型)、uchar (无符号字符型, 默认)、int (整型)、long (长整型)、float (浮

点型)、double (双精度型)等。

- count = fwrite(fid,A,precision,skip): 二进制文件的写入, 参数 skip 控制写入数据 A 的步长, 即数据 A 以一定的间隔写入文件。

【例 10.2】二进制文件的写入。

```
>> fid=fopen('fopentest.txt','w')           %打开文件
fid =
     3
n=fwrite(fid,magic(3),'integer*4')          %二进制文件写入, 并返回写入的数据个数
n =
     9
fclose(fid)
```

函数 fprintf()用于文本文件的写入, 其调用格式如下。

count = fprintf(fid,format,A,...): 在句柄值为 fid 的文件中按照指定的数据格式 format 写入数据 A, 其中数据格式, 由“%[标志][输出最小宽度][.精度][长度]格式符”组成, 其中中括号的内容为可选参数, 标志有-、+、#、0 和空格 5 种; 输出最小宽度, 以十进制整数表示输出数据的最小位数, 若实际位数多于定义的最小位数, 则按实际位数输出, 否则输出数据中少于定义的宽度则补空格或 0; 精度以“.”开头后跟十进制整数, 表示小数的位数或字符的个数, 如果实际维数大于定义的精度数, 则截去超过的部分; 长度包括 h、l 两种, h 表示按短整型量输出, l 表示按长整型量输出; 常见的格式符有 d (整型)、f (浮点型)、s (字符串型)、c (字符型)、e (以指数形式表示的单、双精度实数)等。

【例 10.3】文本文件的写入。

```
fid=fopen('fprintftest.txt','w')
fprintf(fid,'%6f',5.3)
fprintf(fid,'%3e',2)
fclose(fid)
```

文件中将写入: 5.3000002.000000e+000

3. 文件的读出

函数 fread()可用于二进制文件的读出, 并将数据存入 MATLAB 的工作空间内, 其调用格式如下。

- A = fread(fid): 读取句柄值为 fid 的文件内的数据, 并存储到 MATLAB 中的数据矩阵 A 中。
- A = fread(fid, count): 读取句柄值为 fid 的文件内的数据, 参数 count 用于设置读取的数据的个数, 其中参数值可取正整数 n, 代表读取 n 个元素, 取 Inf 代表读取文件内的所有数据, 取[M,N]代表读取 M×N 的矩阵, 存储读取到的数据到矩阵 A 中。
- A = fread(fid, count, precision): 参数 precision 控制读出数据的格式。
- A = fread(fid, count, precision, skip): 参数 skip 与函数 fwrite()中的参数 skip 功能类似, 控制读出数据 A 的步长, 即文件中的数据 A 以一定的间隔读出。
- A = fread(fid, count, precision, skip, machineformat): 参数 machineformat 控制读出数据的精度。
- [A, count] = fread(...): 从文件中读出数据 A, 参数 count 记录读出数据的个数。

【例 10.4】二进制文件的读出。

```
>> fid = fopen('alphabet.txt','r');          %打开文件 alphabet.txt, 其中存放了字母表 A~Z
>> c = fread(fid,5,'*char')                 %以字符格式读取其中的 5 个数据, 返回的数据以列向量的格式存储
c =
A
B
C
D
E
>> c = fread(fid,5,'*char',2)               %以间隔 2 个字符, 读取其中的 5 个数据
```

```

C =
F
I
L
O
R
>> c = fread(fid,5)           %未指定读取的数据为字符，字符变量转换为数值变量
c =
    85
    86
    87
    88
    89
>> fclose(fid)

```

函数 `fscanf()` 用于文本文件的读出，其调用格式如下。

- `A = fscanf(fid,format)`: 以 `format` 的数据格式读取文件句柄值为 `fid` 的文本文件，返回值 `A` 为读取的数据，将存储到 MATLAB 空间中，`format` 格式的设置同函数 `fprintf()`。
- `[A,count] = fscanf(fid,format,size)`: 参数 `size` 用于设置读出的数据大小，`size` 的可选值正整数 `n`，代表 `n` 个数据组成的列向量，或 `[m,n]` 代表 $m \times n$ 的矩阵，参数 `count` 返回读出数据的个数。

【例 10.5】文本文件的读出。

```

x = 0:.1:1;
y = [x; exp(x)];
fid = fopen('fscanrest.txt','w');
fprintf(fid,'%6.2f %12.8f\n',y);           %写入文本文件
fclose(fid)
fid = fopen('fscanrest.txt');
a = fscanf(fid,'%g %g',[2 4])             %读出文本文件
a =
     0     0.1000     0.2000     0.3000
  1.0000  1.1052  1.2214  1.3499

```

另外，MATLAB 还提供了函数 `fgetl()` 和 `fgets()` 用于按行读取数据，其调用格式如下。

- `tline = fgetl(fid)`: 读取句柄为 `fid` 的文件的一行数据，并返回到变量 `tline` 中。
- `tline = fgets(fid)`: 读取句柄为 `fid` 的文件的一行数据，并返回到变量 `tline` 中。
- `tline = fgets(fid,nchar)`: 读取句柄为 `fid` 的文件的一行数据，最多读取 `nchar` 个字符，如果遇到行终止符或者文件末尾则不再读取数据。

函数 `fgetl()` 和 `fgets()` 都可以用于行数据的读取，其中前者在行数据读取时，不读行末尾的回车符号，而函数 `fgets()` 读取字符串末尾的回车符号。

【例 10.6】文本的行读取。

```

>> fid=fopen('mean.m');
tline = fgetl(fid);           %使用函数 fgetl() 读取行数据
disp(tline)                   %显示读取的行数据
function y = mean(x,dim)
>> length(tline)              %读取的行数据的长度
ans =
    24
>> fclose(fid);
>> fid=fopen('mean.m');
tline = fgets(fid);           %使用函数 fgets() 读取行数据
disp(tline)                   %显示读取的行数据
function y = mean(x,dim)
>> length(tline)              %读取的行数据的长度
ans =
    26
>> fclose(fid);

```

4. 文件的关闭

函数 `fclose()` 用于对打开文件执行关闭操作，在对打开的文件执行完读/写等操作后，需要及时关闭。函数 `fclose()` 的调用格式如下。

- `status = fclose(fid)`: 关闭句柄值为 `fid` 的文件，如果关闭成功，将返回值 0，否则返回 -1。
- `status = fclose('all')`: 关闭所有打开的文件。

【例 10.7】文件的关闭。

```
>> fid = fopen('fcloserest.txt','w');
fclose(fid)           %关闭文件成功
ans =
    0
```

5. 文件的定位

函数 `fseek()` 和 `ftell()` 可用于文件的定位，其中函数 `fseek()` 用于定位文件的位置指针，调用格式如下。

- `status = fseek(fid,offset,origin)`: 其中参数 `fid` 为需要定位的文件句柄，参照参数 `origin` 位置指针移到了 `offset` 字节数，`offset` 的值小于 0 向前移动，`offset` 的值大于 0 向后移动，`offset` 的值为 0 不移动，而位置参数 `origin` 的可设置包括 `bof/-1` (文件开头)、`cof/0` (文件当前位置)、`eof/1` (文件末尾)，参数 `status` 返回文件定位的状态，值为 0 表示定位成功，否则返回 -1。

函数 `ftell()` 可用于返回文件指针的当前位置，其调用格式如下。

`position = ftell(fid)`: 返回句柄值为 `fid` 的文件的当前位置 `position`，通过字节数定义文件的当前位置，如果定位失败，返回值为 -1。

函数 `frewind()` 可分别用于移动文件位置指针到文件开头，其调用格式如下。

`frewind(fid)`: 移动句柄值为 `fid` 的文件的位置指针到文件开头的位置。

函数 `feof()` 用来检测位置指针是否在文件的末尾，其调用格式如下。

`eofstat = feof(fid)`: 检测句柄 `fid` 标识的文件的位置指针是否在文件末尾，是则返回状态数为 1，否则返回 0。

【例 10.8】文件的定位。

```
>> fid=fopen('alphabet.txt','r');
fseek(fid,5,-1)       %从文件初始位置向后移动 5 字节
ans =
    0
>> ftell(fid)         %返回文件的当前字节处
ans =
    5
>> fseek(fid,200,'cof') %从当前位置向后移动 200 字节，超过文件范围，定位失败
ans =
   -1
>> fseek(fid,2,1)     %从文件末尾向后移动，定位失败
ans =
   -1
>> frewind(fid)       %文件指针移动到开头
>> ftell(fid)
ans =
    0
>> eofstat = feof(fid) %当前位置指针不在文件末尾
eofstat =
    0
>> fclose(fid)       %关闭文件
ans =
    0
```

6. 文件的存在性检查

函数 `exist()` 用来检查 MATLAB 工作空间和搜索路径下的变量、函数或者文件等是否存在，其调用格式如下。

- `exist item`: 判断名为 `item` 的变量、函数或者文件等是否存在，不存在返回参数 0，存在返回各类型的参数。存在变量，返回 1；存在 M 文件，返回 2；存在 MEX 或 DLL 文件，返回 3；存在 MDL 文件，返回 4；存在内建函数，返回 5；存在 p 代码文件，返回 6；存在目录，返回 7；存在 Java 类，返回 8。
- `exist item kind`: 判断指定类型为 `kind` 的文件、函数或者变量 `item` 是否存在，可设类型包括 `var`（变量）、`file`（文件）、`builtin`（内建函数）和 `dir`（目录）。
- `a = exist('item','kind')`: 返回判断文件是否存在的指示数据到变量 `a`。

【例 10.9】 文件的存在性检查。

```
>> exist fgetl                %判断是否存在 M 文件 fgetl
ans =
     2
>> exist fgetl 'var'         %判断是否存在变量 fgetl
ans =
     0
```

7. 文件输入/输出错误信息的查询

函数 `ferror()` 用于返回文件的输入/输出错误信息，其调用格式如下。

- `message = ferror(fid)`: 句柄值为 `fid` 的文件的输入/输出错误信息。
- `message = ferror(fid,'clear')`: 清除特殊文件标识的错误信息。
- `[message,errnum] = ferror(...)`: 返回错误信息和错误数。

【例 10.10】 文件输入/输出错误信息的查询。

```
>> fid=fopen('alphabet.txt','r');
if fseek(fid,2,1) ~= 0                %如果位置指针无法定位，返回错误信息
    msg = ferror(fid);
    disp(msg);
end
Offset is bad - after end-of-file or last character written.
```

10.1.2 高级文件的 I/O 操作

前面一节介绍的低级文件的 I/O 操作函数与 C 语言的函数类似，而 MATLAB 也为用户提供了高级文件的 I/O 操作的函数，这些函数使用更为方便。本节将重点讲述高级文件的 I/O 操作函数的使用，其中部分函数在前面的章节中已有所涉及，这里将更为全面地把函数的功能仔细叙述一遍，同时便于读者与类似的函数进行比较。

1. 文件的打开

函数 `open()` 可用于打开 MATLAB 支持的各种文件，在打开的过程中 MATLAB 将会根据文件的扩展名选择相应的编辑窗口打开文件，函数的调用格式如下。

`open('name')`: 打开文件名为 `name` 的文件，`name` 需带文件后缀。

同时，函数 `open()` 可以打开 MAT 格式的数据文件，但多个变量下会以结构体的方式打开。

【例 10.11】 利用函数 `open()` 打开文件。

```
open('sort.m')                %打开 M 文件
open('cars.mat')              %以结构体的形式打开多个数据变量的数据文件
open('cars2.mat')             %以矩阵的形式打开单个数据变量的数据文件
```

2. 数据的导入

MAT 格式是 MATLAB 数据的主要存储格式,该格式能较好地适用于 MATLAB 平台。常用的 MAT 格式数据文件的导入函数如下。

1) 函数 load()

函数 load()用于导入 MATLAB 搜索路径或指定路径下的 MAT、二进制、ASCII 格式的数据文件,其调用格式如下。

- load: 导入文件 matlab.mat 中的所有变量到 MATLAB 工作空间,如果不存在,则返回错误信息。
- load('filename'): 导入文件名为 filename 中的全部变量到 MATLAB 工作空间, filename 可为包含指定路径的文件名,或为在 MATLAB 搜索路径下的文件名,如果 filename 不带文件后缀,则将以二进制方式,导入“MAT”格式的数据文件,如果 filename 带“MAT”以外的文件后缀,则将以 ASCII 格式导入该文件中的所有数据。
- load('filename', 'X', 'Y', 'Z'): 从文件名为 filename 的 MAT 数据文件中导入变量 X、Y、Z。
- load('-mat', 'filename'): 不论文件名 filename 是否带扩展名,数据都将以 MAT 格式导入,如果需要导入的文件不是 MAT 文件,则将返回导入错误的信息。
- load('-ascii', 'filename'): 不论文件名 filename 是否带扩展名,数据都将以 ASCII 格式导入,如果需要导入的文件不是 ASCII 文件,则将返回导入错误的信息。
- S = load(...): 导入数据存储到变量 S 中,如果导入的为 MAT 文件,则变量 S 将为结构体变量,各域中存储 MAT 文件的各数据变量,如果为 ASCII 文件,则导入的数据将以双精度的数据矩阵存储。
- load filename -regexp expr1 expr2 ...: 通过正则表达式控制需要导入的变量。

【例 10.12】利用函数 load()导入数据。

```
clear
load cars                %导入当前路径下的变量
clear
load C:\MATLAB701\work\10\cars %导入指定路径下的数据文件
clear
load('cars', 'Acceleration') %导入数据文件中的指定变量
clear
load('-mat', 'cars')         %指定导入 MAT 格式的数据文件
clear
load('-ascii', 'cars')      %指定导入 ASCII 格式的数据文件,文件非 ASCII 文件,导入出错
??? Error using ==> load
Number of columns on line 2 of ASCII file C:\MATLAB701\work\10\cars.mat
must be the same as previous lines.
```

2) 函数 importdata()

MATLAB 中函数 importdata()也常用于数据导入,其功能界面操作的数据导入平台 Import Wizard 功能类似,导入的数据如果有多个变量将会以结构体的形式存储,单个变量将会以矩阵的形式存储;同时如果数据的中间有字符内容,将无法导入全部数据,即数据读入到字符串处将停止数据的读入,但如果字符串出现在文件开头,将不影响数据的导入工作。函数 importdata()的调用格式如下。

- importdata('filename'): 将 filename 文件中的数据导入到 MATLAB 工作区中,文件名需带扩展名。
- A = importdata('filename'): 将 filename 文件中的数据导入到变量 A 中,并存储到 MATLAB 工作区中。
- importdata('filename','delimiter'): 将 filename 文件中的数据导入,参数 delimiter 设置导

入数据的分隔符, 如果需要 Tab 键, 可以用字符 t 指定。

【例 10.13】利用函数 importdata() 导入数据。

```
clear
a=importdata('cars.mat');           %导入数据文件 cars.mat, 其中存储多个变量, 返回结果 a 为结构体
clear
a=importdata('cars2.mat');          %导入数据文件 cars2.mat, 其中存储 1 个变量, 返回结果 a 为矩阵
3) 函数 textread()
```

函数 importdata() 无法导入在文件中间同时含有字符和数值的文件, 而对于此类文件适合使用函数 textread() 导入。函数 textread() 主要用于 txt 文件的导入, 其使用方法如下。

- [A,B,C,...] = textread('filename','format'): 读取文件名为 filename 的数据文件, 参数 format 用于设置读取的数据文件的格式, 需对每个读出的数据设置格式, 返回读取的数据于变量 A、B、C... 中。
- [A,B,C,...] = textread('filename','format',N): 参数 N 用于设置读取的数据行数。
- [...] = textread(...,'param','value',...): 设置读取数据的特殊格式, 具体参见帮助文档。

【例 10.14】利用函数 textread() 导入数据。

新建 txt 文本, 在其中写入 “Sally Level1 12.34 45 Yes”

```
[names, types, y, answer] = textread('textreadtest.txt', '%9c %5s %*f %2d %3s', 1)
    %读出第一行数据
names =
Sally
types =
'Level'
y =
12
answer =
'.34'

>> A = textread('text.txt')           %读出文本内的数据
A =
    0.9501    0.4565    0.9218    0.4103    0.1389    0.0153
    0.2311    0.0185    0.7382    0.8936    0.2028    0.7468
    0.6068    0.8214    0.1763    0.0579    0.1987    0.4451
    0.4860    0.4447    0.4057    0.3529    0.6038    0.9318
    0.8913    0.6154    0.9355    0.8132    0.2722    0.4660
    0.7621    0.7919    0.9169    0.0099    0.1988    0.4186
```

4) 函数 textscan()

函数 textscan() 也可快速读取文本数据, 其功能与函数 textread() 类似, 但是使用函数 textscan() 导入数据需要先打开文件, 获取文件句柄, 并且读入的数据保存为单元数组。函数 textscan() 的调用格式如下。

- C = textscan(fid, 'format'): 以格式 format 读取句柄值为 fid 的文本文件。
- C = textscan(fid, 'format', N): 参数 N 用于指定读取数据的次数。

【例 10.15】利用函数 textscan() 导入数据。

```
fid=fopen('text.txt','r')
A = textscan(fid,'%f32')
fclose(fid)
```

5) 函数 csvread()

函数 csvread() 主要用于读取以逗号分隔的文本数据, 其调用格式如下。

- M = csvread('filename'): 读入文件 filename 内的文本数据, 数据间以逗号分隔, 读入的数据存储于矩阵 M 中。
- M = csvread('filename', row, col): 指定读取数据的行 row 和列 col, 第一行和第一列的数据的行列号为 0 和 0。

- `M = csvread('filename', row, col, range)`: 参数 `range` 用于指定读取数据的范围, 使用向量 `[R1 C1 R2 C2]` 表示, 其中 `R1` 和 `C1` 为左上角的行列号, `R2` 和 `C2` 为右下角的行列号。

【例 10.16】 利用函数 `csvread()` 导入数据。

```
>> M = csvread('csvtest.dat') %读取以逗号分隔的文本数据
M =
    35     1     6    26    19    24
     3    32     7    21    23    25
    31     9     2    22    27    20
     8    28    33    17    10    15
    30     5    34    12    14    16
     4    36    29    13    18    11

>> M = csvread('csvtest.dat', 2, 3) %指定读取数据的范围
M =
    22    27    20
    17    10    15
    12    14    16
    13    18    11
```

6) 函数 `dlmread()`

函数 `dlmread()` 可用于读取带分割的 ASCII 数据文件, 并可指定读入数据的范围, 其调用格式如下。

- `M = dlmread('filename')`: 导入文件名为 `filename`, 默认的数据分隔符为逗号 “,”, 导入的数据返回到矩阵 `M`。
- `M = dlmread('filename', delimiter)`: 导入分隔符为 `delimiter` 的数据文件 `filename`, 使用 “\t” 代表 Tab 分隔符。
- `M = dlmread('filename', delimiter, R, C)`: 参数 `R` 和 `C` 分别代表了要读取的数据在文件中左上角的行列号, `R=0`, `C=0` 代表第一个元素。
- `M = dlmread('filename', delimiter, range)`: 参数 `range` 用于指定读取数据的范围, 使用向量 `[R1 C1 R2 C2]` 表示, 其中 `R1` 和 `C1` 为左上角的行列号, `R2` 和 `C2` 为右下角的行列号。

【例 10.17】 利用函数 `dlmread()` 导入数据。

```
>> M = dlmread('dlmtest.txt')
M =
    0.9501    0.7621    0.6154    0.4057    0.0579
    0.2311    0.4565    0.7919    0.9355    0.3529
    0.6068    0.0185    0.9218    0.9169    0.8132
    0.4860    0.8214    0.7382    0.4103    0.0099
    0.8913    0.4447    0.1763    0.8937    0.1389
    0.2028    0.2722    0.7468         0         0
    0.1987    0.1988    0.4451         0         0
    0.6038    0.0153    0.9318         0         0

>> M = dlmread('dlmtest.txt', '\t', 2, 3) %读出指定范围的数据
M =
    0.9169    0.8132
    0.4103    0.0099
    0.8937    0.1389
         0         0
         0         0
    0.2722    0.7468
    0.1988    0.4451
    0.0153    0.9318
```

7) 函数 `xlsread()`

对于一般用户可能 Excel 还是主要的数据存储文件, Excel 文件操作简便、快捷、灵活。函数 `xlsread()` 可用于直接从 Excel 中读取数据, 其调用格式如下。

- `N = xlsread('filename')`: 读取 Excel 文件 `filename` 中的数据, 并存储到矩阵 `N` 中。

- `N = xlsread('filename', -1)`: 该格式将打开 `filename` 文件, 提示用户选择需要导入的数据, 选择的数据将导入矩阵 `N` 中。
- `N = xlsread('filename', sheet)`: 用于打开指定工作簿 `sheet` 中的数据到矩阵 `N`。
- `N = xlsread('filename', 'range')`: 导入指定范围 `range` 的数据到矩阵 `N` 中。
- `N = xlsread('filename', sheet, 'range')`: 打开指定工作簿 `sheet`、指定范围 `range` 的数据到矩阵 `N` 中。
- `[N, T] = xlsread('filename', ...)`: 读取 Excel 文件 `filename` 中的数据, 其中数据文件存储到变量 `N` 中, 文本文件存储到变量 `T` 中。
- `[N, T, rawdata] = xlsread('filename', ...)`: 读取 Excel 文件 `filename` 中的数据, 其中数据文件存储到变量 `N` 中, 文本文件存储到变量 `T` 中, 变量 `rawdata` 中同时存储数据和文本文件。

【例 10.18】 利用函数 `xlsread()` 导入数据。

```
>> N = xlsread('xlstest.xls')    %读取 Excel 中的数据, 返回到变量 N, 中间的空格读入后为 NaN
N =
    0.4103    0.3529    0.1389
    0.8936    0.8132    0.2028
    0.0579    0.0099    0.1987
         NaN         NaN         NaN
         NaN         NaN         NaN
         NaN         NaN         NaN
         NaN         NaN         NaN
   12.0000   98.0000         NaN
   13.0000   99.0000         NaN
   14.0000   97.0000         NaN
N = xlsread('xlstest.xls', -1)    %打开 Excel 文件, 提示用户选择数据, 选择后单击如图 10.1 中
                                   %所示的“OK”按钮
                                   %选择的数据导入到变量 N 中
N =
    0.4103    0.3529    0.1389
    0.8936    0.8132    0.2028
    0.0579    0.0099    0.1987
>> [N, T] = xlsread('xlstest.xls') %导入数据, 数字数据存储到变量 N 中, 文本数据存储到变量 T
N =
    0.4103    0.3529    0.1389
    0.8936    0.8132    0.2028
    0.0579    0.0099    0.1987
         NaN         NaN         NaN
         NaN         NaN         NaN
         NaN         NaN         NaN
         NaN         NaN         NaN
   12.0000   98.0000         NaN
   13.0000   99.0000         NaN
   14.0000   97.0000         NaN
T =
    'Time'    'Temp'
```

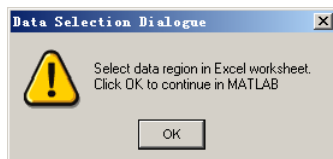


图 10.1 Excel 数据区域选择

3. 数据的导出

1) 函数 `save()`

函数 `save()` 是最为常用的数据导出函数, 可将 MATLAB 工作空间内的变量保存到外部磁盘上,

以供下次使用的时候导入。函数 `save()` 保存的数据格式可以为 MAT 文件、ASCII 文件等，与函数 `load()` 的导入功能可以配合使用。

函数 `save()` 的调用格式如下。

- `save`: 将 MATLAB 工作空间内的所有变量保存到文件 “matlab.mat” 中，MAT 格式的数据文件为 MATLAB 保存数据的主要格式，在 MATLAB 中的导入导出也比较方便、高效。
- `save('filename')`: 保存 MATLAB 工作空间内的所有变量到名为 `filename` 的 MAT 格式文件中，如果 `filename` 为带指定路径的文件名，则数据文件保存到指定路径下，否则保存在当前路径下。
- `save('filename', 'var1', 'var2', ...)`: 保存指定的变量 `var1`、`var2`... 到数据文件 `filename` 中。
- `save('filename', '-struct', 's')`: 保存结构体变量 `s` 到名为 `filename` 的 MAT 格式文件中，同时结构体 `s` 中的各域独立存储。
- `save('filename', '-struct', 's', 'f1', 'f2', ...)`: 保存结构体变量 `s` 中的变量 `f1`、`f2`... 到数据文件 `filename` 中。
- `save('-regexp', expr1, expr2, ...)`: 通过正则表达式保存指定的变量。
- `save('...', 'format')`: 以指定的格式 `format` 保存数据变量，包括的格式有 `-append` (在已经存在的 mat 文件中添加此变量)、`-ascii` (8 位 ASCII 格式)、`-ascii -tabs` (以 tab 位分隔的 8 位 ASCII 格式)、`-ascii -double` (16 位 ASCII 格式)、`-ascii -double -tabs` (以 tab 位分隔的 16 位 ASCII 格式) 和 `-mat` (默认的 MAT 格式)。
- `save filename var1 var2 ...`: 保存指定的变量 `var1`、`var2`... 到数据文件 `filename` 中。

【例 10.19】 利用函数 `save()` 导出数据。

```
clear
load cars                %导入数据变量
save                     %MATLAB 工作空间内的所有变量保存为 MAT 格式的文件 matlab.mat
save cars2               %保存当前工作空间内的变量到文件 cars2.mat 中
save cars2 Acceleration %保存指定变量到数据文件中
save('cars2.txt', '-ascii') %保存变量为 ASCII 格式的数据文件
```

执行上述程序，在 MATLAB 的当前路径下将会新增这些数据文件，下次需要使用的时候可以直接通过函数 `load()` 调用。

2) 函数 `csvwrite()`

函数 `csvwrite()` 用于向文本文件中写入以逗号分隔的数据，其调用格式如下。

- `csvwrite('filename', M)`: 向文本文件 `filename` 中写入矩阵 `M`，数据间以逗号分隔。
- `csvwrite('filename', M, row, col)`: 写入逗号分隔的数据，并知道起始的行列号，其中规定数据的第一行、第一列 (`row=0`、`col=0`)。

【例 10.20】 利用函数 `csvwrite()` 导出数据。

```
a=magic(6);
csvwrite('csvtest.dat', a)

写入数据的文件 csvtest.dat 中的内容。

35,1,6,26,19,24
3,32,7,21,23,25
31,9,2,22,27,20
8,28,33,17,10,15
30,5,34,12,14,16
4,36,29,13,18,11

csvwrite('csvtest2.dat', a, 2, 1) %指定写入数据的行列号
```

写入数据的文件 `csvtest2.dat` 中的内容。

```
''''''
''''''
```

```
,35,1,6,26,19,24
,3,32,7,21,23,25
,31,9,2,22,27,20
,8,28,33,17,10,15
,30,5,34,12,14,16
,4,36,29,13,18,11
```

3) 函数 dlmwrite()

函数 dlmwrite() 可用于向文件中写入指定分隔符的数据，其调用格式如下。

- dlmwrite('filename', M): 在文件 filename 中写入数据矩阵 M，默认状态下写入的数据以逗号分隔。
- dlmwrite('filename', M, 'D'): 采用指定的分隔符 D 写入数据文件 M 到文件 filename。
- dlmwrite('filename', M, 'D', R, C): 写入数据 M，并指定数据的起始行 R 和起始列 C，都是从 0 开始。
- dlmwrite('filename', M, attribute1, value1, attribute2, value2, ...): 导出数据，并设置相关属性，包括 newline (指定换行符)、delimiter (指定分隔符)、roffset (行偏移，即文件第一行的位置，起始为 0)、coffset (列偏移，即文件第一列的位置，起始为 0) 和 precision (指定精确度)。
- dlmwrite('filename', M, '-append'): 以添加方式写入数据，即在原文件的后面写入数据，不覆盖原文件。
- dlmwrite('filename', M, '-append', attribute-value list): 在文件末尾写入数据，并设置相关属性。

【例 10.21】 利用函数 dlmwrite() 导出数据。

```
A=rand(5);
dlmwrite('dlmtest.txt', A, 'delimiter', '\t', 'precision', 6)
```

写入的数据如下：

```
0.950129    0.762097    0.615432    0.405706    0.0578913
0.231139    0.456468    0.791937    0.93547    0.352868
0.606843    0.0185036    0.921813    0.916904    0.813166
0.485982    0.821407    0.738207    0.41027    0.0098613
0.891299    0.444703    0.176266    0.89365    0.138891
```

在文件末尾添加数据后：

```
A=rand(3);
dlmwrite('dlmtest.txt', A, '-append', 'delimiter', '\t', 'precision',
6, 'roffset', 2, 'coffset', 2)
```

添加数据后文件内容如下：

```
0.950129    0.762097    0.615432    0.405706    0.0578913
0.231139    0.456468    0.791937    0.93547    0.352868
0.606843    0.0185036    0.921813    0.916904    0.813166
0.485982    0.821407    0.738207    0.41027    0.0098613
0.891299    0.444703    0.176266    0.89365    0.138891
```

```
0.202765    0.272188    0.746786
0.198722    0.198814    0.445096
0.603792    0.0152739    0.931815
```

4) 函数 xlswrite()

函数 xlswrite() 用于向 Excel 中写入数据，其调用格式如下。

- xlswrite('filename', M): 将数据 M 写入 Excel 文件 filename 中。
- xlswrite('filename', M, sheet): 在指定的工作簿 sheet 中写入数据 M。
- xlswrite('filename', M, 'range'): 在指定的范围内写入数据 M，参数 range 可以是起始的单

元格或者单元格范围，例如 A1（在单元格 A1 中开始写数据）或者 A1:D3（写入数据的范围在单元格 A1:D3）。

- `xlswrite('filename', M, sheet, 'range')`: 在指定的工作簿 `sheet` 写入数据，并指定写入数据的范围。
- `status = xlswrite('filename', ...)`: 向 Excel 中写入数据，并返回写入数据的状态 `status`，值为 1 代表写入成功，值为 0 代表写入不成功。
- `[status, message] = xlswrite('filename', ...)`: 向 Excel 中写入数据，并返回写入数据的状态 `status` 和写入数据时的警告和错误信息 `message`。
- `xlswrite filename M sheet range`: 向 Excel 中指定范围写入数据的另一种格式表示。

【例 10.22】 利用函数 `xlswrite()` 导出数据。

```
a=rand(3);
xlswrite('xlstest.xls', a, 'sheet1', 'A2')           %向 Excel 中写入数据文件
d = {'Time', 'Temp'; 12 98; 13 99; 14 97}
xlswrite('xlstest.xls', d, 'sheet1', 'A8')           %向 Excel 中写入数据和字符串
向 Excel 中写入数据后如图 10.2 所示。
```


	A	B	C
1			
2	0.41027	0.352868	0.138891
3	0.89365	0.813166	0.202765
4	0.057891	0.009861	0.198722
5			
6			
7			
8	Time	Temp	
9	12	98	
10	13	99	
11	14	97	
12			

图 10.2 Excel 中导入的数据

10.1.3 利用界面工具导入/导出数据

在前面的章节中介绍了如何使用函数导入/导出数据，MATLAB 同时也为用户提供了界面工具可以更为直观地导入/导出数据。


数据导入界面的方式有如下几种。

- 通过菜单命令，选择主界面“File”→“Import data”命令，直接选择需要导入的数据。
- 利用 Workspace 窗口的工具，导入数据。
- 通过函数命令 `uiimport`，打开数据导入平台。
- 通过 MATLAB 的 Start 菜单中“Matlab”→“Import wizard”菜单项，打开数据导入平台，导入数据。

下面以具体的实例演示数据的导入。

【例 10.23】 利用界面工具导入数据。

本实例将介绍如何利用数据导入界面成功导入数据，其中上面介绍的数据导入方法中，前两种方式导入过程类似，将在本实例中的 1、2 两点中演示其使用，而导入的后两者方法将在实例中第 3 点中展开叙述。通过本实例的演示，读者可以掌握界面工具导入数据的技巧，并比较各方法的差异，在今后运行中灵活选用各方法。

(1) 选择“File”→“Import data”命令或者 Workspace 窗口的工具，将打开如图 10.3 所示的数据文件打开对话框。选择需要打开的数据文件“cars.mat”，单击“打开”按钮。

(2) 进入如图 10.4 所示的数据导入平台，显示即将导入的数据的结构，此实例中导入的数据

包含 4 个变量,同时双击导入的变量,还可以在数据导入平台右侧具体显示变量的数值,如图 10.5 所示。此时,用户可以检查导入的数据是否有误,有误的话可以单击“Back”按钮,重新导入数据,如果确认无误,导入的数据将存储到 MATLAB 工作空间内。

(3) 通过函数命令 `uiimport` 或者 Start 菜单中“Matlab”→“Import wizard”菜单项将可直接打开如图 10.6 所示的数据导入平台。其中,可以选择通过“File”或者“Clipboard”导入数据,本实例中选择通过文件导入数据,单击界面中的“Browse”按钮浏览选择需要打开的数据文件,选择后数据的基本信息将显示在 Import wizard 的左侧区域,同时右侧区域的各选项卡分别显示数据文件的各变量。最后,单击“Finsh”按钮,数据导入到 MATLAB 工作空间内。

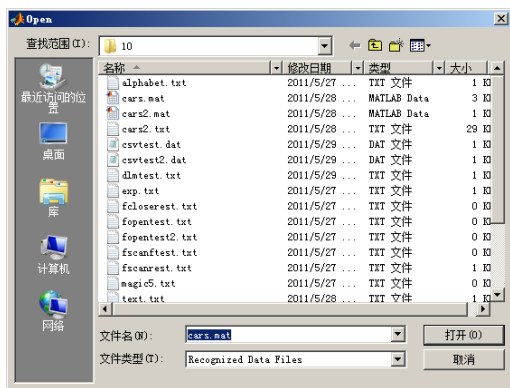


图 10.3 数据文件打开对话框

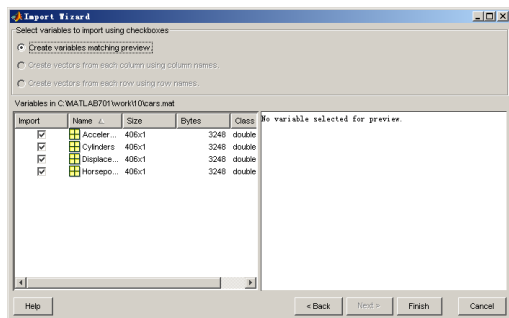


图 10.4 数据导入平台

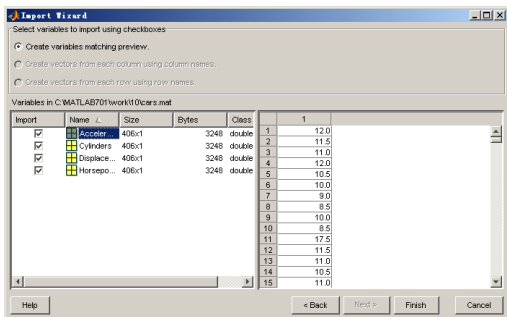


图 10.5 数据导入平台数据显示

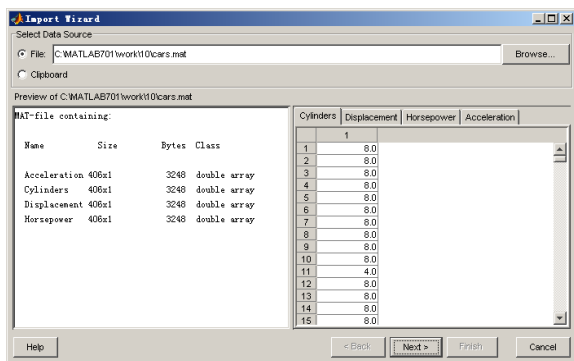


图 10.6 数据导入平台

MATLAB 并没有提供专门的窗口用于数据的导出,但是我们可以可以在菜单中通过一些命令实现数据的保存。具体方法如下。

- 通过菜单命令,选择“File”→“Save workspace as”命令,直接选择需要导出的数据。
- 利用 Workspace 窗口的 工具或右键快捷菜单中的“Save As”命令,导出数据。
- 数据编辑窗口的 工具。
- 在数据编辑窗口中直接选择需要的数据,然后复制,粘贴。

其中,前 3 种方法只可以导出 MAT 格式的数据文件,而最后一种方式类似于 Excel 中的数据复制/粘贴操作,用户只需要在打开的变量数据编辑窗口选中需要保存的数据,复制后可以直接粘贴到 TXT、Excel 等数据存储文件中,同时也可以把 Excel 中的数据直接复制到数据编辑器。而同时对于 MAT 格式数据的保存,当不选中任何变量时,前两种方法将把工作空间内的所有变量全部保存下来,默认下保存在文件“matlab.mat”中,而第三种方法仅保存当前数据编辑窗口的变量。

另外,用户还可以通过函数 `uigetfile()` 打开文件对话框,从文件对话框中选择需要打开的文件,

利用该函数将返回用户选择打开的文件名，具体文件的打开还需要使用相关的文件打开的函数，类似的函数 `uiputfile()` 可用于打开文件的保存对话框，执行文件保存命令。关于这两个函数的使用在 GUI 编程的章节中有所涉及，这里不再详细展开叙述。

10.2 图片文件

前面一小节中详细介绍了数据文件的导入/导出，而在 MATLAB 中图片文件也很重要，如何导入/导出图片文件也是很多用户关心的。其中通过界面操作在图形窗口中打开或者保存图片文件的操作比较简单，类似于一般的 Office 操作，在前面的章节中也有所涉及，这里不再详细展开叙述。本章将向读者介绍如何通过函数把不同格式图片文件的导入和导出。

10.2.1 不同格式图片文件的导入

在 MATLAB 中提供了函数 `imread()` 用于导入不同格式的图片文件，其调用格式如下。

- `A = imread(filename,fmt)`: 读取文件名为 `filename` 的图片文件，参数 `fmt` 指定读取图片的类型，导入的图片存储到变量 `A` 中。
- `[X,map] = imread(filename,fmt)`: 导入指定格式的图片文件，返回图像的数据矩阵 `X` 和颜色映射表 `map`。
- `[...] = imread(filename)`: 直接读取图片文件，自动识别文件的类型。
- `[...] = imread(URL,...)`: 读取网络中的图片，参数 `URL` 必须为格式 “`http://....`”。
- `[...] = imread(...,idx)`: 只适用于读取 CUR、GIF、ICO 和 TIFF 格式的图片文件。
- `[...] = imread(...,'PixelRegion',{ROWS, COLS})`: 该格式只适用于读取 TIFF 格式的图片文件。
- `[...] = imread(...,'frames',idx)`: 该格式只适用于读取 GIF 格式的图片文件。
- `[...] = imread(...,ref)`: 该格式只适用于读取 HDF 格式的图片文件。
- `[...] = imread(...,'BackgroundColor',BG)`: 该格式只适用于读取 PNG 格式的图片文件。
- `[A,map,alpha] = imread(...)`: 该格式只适用于读取 ICO、CUR 和 PNG 格式的图片文件。

一般情况下，由于读取的图像文件多为 8 位的，所以存储图像文件的数据矩阵为 `unit8` 数据类型。

【例 10.24】图片文件的导入。

```
A = imread('abc.jpeg','jpg');           %导入 jpg 的图片文件
B = imread('abc2.tif');                 %直接读取图片文件，MATLAB 会根据文件后缀直接判断文件类型
读取的图片将会以数据矩阵的形式存储在 MATLAB 中，数据为图像长方向像素×宽方向像素×3。
```

10.2.2 不同格式图片文件的导出

函数 `imwrite()` 用于对数据形式存储的图片文件进行保存操作，与函数 `imread()` 相对使用。该函数的调用格式如下。

- `imwrite(A,filename,fmt)`: 导出数据 `A` 所代表的图像文件，`filename` 为保存的文件名，`fmt` 为保存的文件格式。
- `imwrite(X,map,filename,fmt)`: 导出数据矩阵为 `X` 的索引图像，参数 `map` 为其演示映射表。
- `imwrite(...,filename)`: 导出图片文件，根据文件名的扩展识别保存的文件类型。
- `imwrite(...,Param1,Val1,Param2,Val2...)`: 导出图片文件，并设置相关参数，对于不同格式的图片文件可以设置不同的属性，具体参考帮助文档。

另外, 函数 `print()` 和 `saveas()` 也可以用于图片的保存, 在本书第 5 章相关章节已介绍, 这里不再详细展开叙述, 仅在实例中演示其使用。

【例 10.25】 图片文件的导出。

```
x=0:pi/100:2*pi;
plot(sin(x));
print(gcf, '-djpeg', 'abc.jpeg');           %通过打印方式保存图片
plot(sin(2*x));
saveas(gca, 'abc2.jpeg', 'jpg');           %通过另存为方式保存图片
saveas(gca, 'abc2.tif', 'tif');
A = imread('abc.jpeg', 'jpg');
imwrite(A, 'abc3.jpeg', 'jpg');           %通过函数 imwrite() 保存图片
```

10.3 本章小结

本章主要介绍了 MATLAB 数据文件和图片文件的导入/导出, 在介绍数据文件的导入/导出时, 介绍了大量的函数, 读者在实际应用中可以根据实际导入/导出数据特征选择相应的函数, 而一般图像文件的导入/导出如果用户没有特别的要求, 在界面中利用菜单操作是最为方便的, 但是如果用户需要自动在程序运行中打开或保存生成的图像文件, 可以利用本章中介绍的函数。

本章全面而系统地介绍了 MATLAB 导入/导出函数, 通过本章的学习, 读者将掌握数据文件、图像文件的导入/导出, 再结合前面章节的学习, 读者可以尝试编写相应的代码。

第 2 篇

常用工具箱使用

- 第 11 章 Simulink 仿真
- 第 12 章 统计工具箱
- 第 13 章 图像处理工具箱
- 第 14 章 优化工具箱
- 第 15 章 曲线拟合工具箱
- 第 16 章 神经网络工具箱
- 第 17 章 金融工具箱
- 第 18 章 小波分析工具箱
- 第 19 章 遗传算法工具箱
- 第 20 章 MATLAB 在各领域的应用



第11章

Simulink 仿真

Simulink 是 MATLAB 的重要组成部分，可用于动态地建模、仿真、分析。对于一些做实验无法获得准确结果或者实验代价过高的问题，可以使用 Simulink 的仿真功能，模拟仿真线性或者非线性，连续的或者离散的系统，对系统一定输入条件下的可能输出结果进行分析。Simulink 图形化的操作特点，使其易学、易掌握，在控制仿真界得到了广泛应用。

本章将向读者介绍 Simulink 的基础知识，包括 Simulink 特点、Simulink 建模环境、Simulink 模型库、Simulink 建模仿真的实现。通过本章的学习将使读者初步掌握 Simulink 的使用，建立小型的系统，进行建模仿真分析。

11.1 Simulink 简介

Simulink 是 MATLAB 专门为用户提供的控制系统建模仿真的工具，通过建立系统仿真模型、设置不同的仿真参数、选择不同的输出方式来分析观察仿真结果等功能。Simulink 提供了大量的模块库，基本能满足用户的需要。同时，Simulink 中的模块库是按照图形化的原理设计的，用户无须详细了解模块的内部细节，只需要知道模块的输入/输出，即可通过鼠标的单击与拖曳完成仿真模型的建立。Simulink 模块具有较好封装性的同时，用户也可以设置不同的参数，以仿真不同的模型。

Simulink 功能强大，本节将主要从 Simulink 特点、Simulink 工作环境两方面对 Simulink 的基础知识做简单介绍，通过本章的学习，读者将能对 Simulink 及其操作环境有基本的了解。

11.1.1 Simulink 特点

1. 交互性强

Simulink 仿真开发采用交互式的方法，操作简单、直观，用户只需要通过鼠标直接拖放各模块，动态链接各模块，并为各模块设置仿真参数，即可实现动态系统仿真。图形化的仿真界面可以有效地避免过多的编程，同时也能更为直观地反映仿真的过程，用户可以把主要精力放在仿真的过程中。

2. 功能强

Simulink 是基于 MATLAB 的图形化建模仿真工具，因而，Simulink 与 MATLAB 其他功能函数有较好的交互性，在 Simulink 中可以方便地使用 MATLAB 强大的数值分析和图像处理等功能，可以方便地分析 Simulink 仿真得到的结果，并以图形化的方式显示。同时，Simulink 中集成了大量的模块库，供用户直接使用。模块库涵盖了系统仿真模块库、通信模块库、数字信号处理模块库、控制系统模块库等各种用途的模块库。功能强大的模块库可以满足广大用户的各种要求。

3. 扩展性强

Simulink 具有较好的扩展性, 用户可以根据需要编写自己的模块库, 建立子系统, 封装自定义的模块。对于特别复杂的系统用户也可以编写 M 函数实现仿真建模, 同时支持与其他软件 (C、Fortan 等) 无缝链接。


4. 灵活性强


Simulink 是一款相当灵活的模拟仿真工具, 虽然大部分的模块 MATLAB 已为用户封装好, 但是用户在使用的时候可以方便地改变模块的参数。近年来, 在通信系统、电力系统、信号处理等领域都得到了广泛的应用。同时, 对于一些复杂系统, 图形化操作可能不是很方便, Simulink 也提供了命令行的方式。

11.1.2 Simulink 工作环境

1. Simulink 启动

Simulink 在每次运行 MATLAB 后, 并不会自动打开, 需要用户启动。启动 Simulink 的方法有如下 3 种。

- 单击主界面  工具, 启动 Simulink。
- 选择 “Start” → “Simulink” → “Simulink Library Browser” 命令, 启动 Simulink。
- 在 MATLAB 命令窗口中直接输入 simulink 命令, 将启动 Simulink, 显示 Simulink Library Browser 窗口。

通过上述方式启动 Simulink 后, 可打开如图 11.1 所示的 “Simulink Library Browser” 窗口。选择主界面 “File” → “New” → “Model” 命令, 或者单击 “Simulink Library Browser” 窗口工具栏上的  图标, 可打开 Simulink 编辑窗口, 新建仿真程序, 如图 11.2 所示。用户可以拖动 “Simulink Library Browser” 窗口中的模块到 Simulink 编辑窗口, 建立模拟仿真程序。在下面的内容中将详细介绍 Simulink 的这两个窗口。

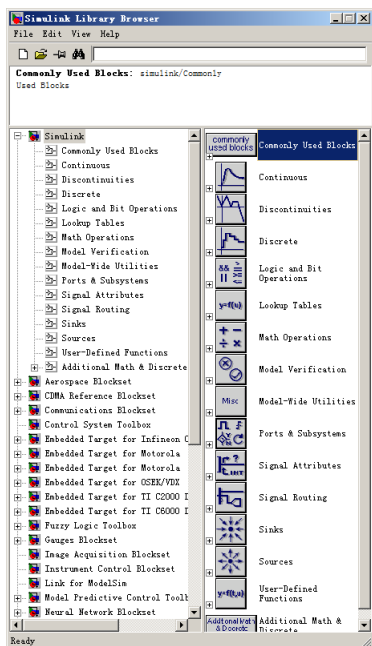


图 11.1 “Simulink Library Browser” 窗口

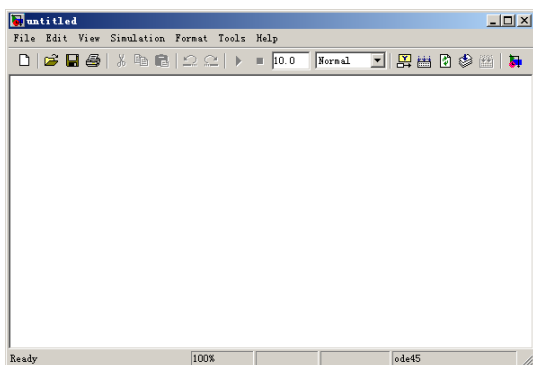


图 11.2 Simulink 编辑窗口

2. “Simulink Library Browser” 窗口

“Simulink Library Browser” 窗口由菜单栏、工具栏和模块工作区组成。其中，菜单栏包括“File”、“Edit”、“View”、“Help”菜单项，这里主要介绍菜单“Edit”和“View”，其他的菜单项与本书之前介绍的类似。

(1) “Edit” 菜单如图 11.3 所示，主要包含如下几个菜单项。

- “Add to the current model” 菜单项：可将“Simulink Library Browser”窗口中选中的模块添加到模型编辑窗口。
- “Find block” 菜单项：用于根据用户提供的关键词在 Simulink 模块库中查找相关的模块，并显示找到的第一个模块。
- “Find next block” 菜单项：继续查找下一个模块。

Add to the current model	Ctrl+I
Find block...	Ctrl+F
Find next block	F3





图 11.3 “Simulink Library Browser” 窗口的“Edit”菜单

(2) “View” 菜单主要用于控制“Simulink Library Browser”窗口各工具的显示，如图 11.4 所示，主要包含如下几个菜单项。

- “Toolbar” 菜单项：控制“Simulink Library Browser”窗口工具条的显示。
- “Status bar” 菜单项：控制“Simulink Library Browser”窗口状态栏的显示。
- “Description” 菜单项：控制“Simulink Library Browser”窗口对选中的模块的描述信息的显示。
- “Stay on top” 菜单项：保持建模过程中“Simulink Library Browser”窗口在界面的最上方。
- “Collapse entire Browser” 菜单项：控制“Simulink Library Browser”窗口左侧的模块浏览器窗口以合并的树形结构形式浏览各模块库。
- “Expand entire Browser” 菜单项：控制“Simulink Library Browser”窗口左侧的模块浏览器窗口以扩展的树形结构形式浏览各模块库。
- “Large icons” 菜单项：以大图标形式浏览各模块。
- “Small icons” 菜单项：以小图标形式浏览各模块。
- “Show Parameters for selected block” 菜单项：显示各模块的参数设置。

✓ Toolbar
✓ Status bar
✓ Description
Stay on top
Collapse entire Browser
Expand entire Browser
✓ Large icons
Small icons
Show Parameters for selected block

图 11.4 “Simulink Library Browser” 窗口的“View”菜单

工具栏中包括了  (新建 Simulink 模型)、 (打开已有的 Simulink 模型)、 (控制窗口在最上方)、 (查找指定的模块)。

模块工作区由 3 部分组成，位于工具栏下方的矩形区域为显示选中的模块描述信息，下方的左侧文件树形区域为系统模块库的浏览，单击相应的系统模块库在右侧的区域将显示该模块库的所有模块。选中需要的模块，可以直接拖动到 Simulink 模型编辑窗口。

综上，可以看到“Simulink Library Browser”窗口为用户提供了方便地浏览、管理各模块的功能，使用该窗口用户可以方便地查找需要的模块，并获得模块简单的描述，同时需要的模块可以直接添加到 Simulink 模型中，操作简单、方便。

3. Simulink 模型编辑窗口

Simulink 模型编辑窗口包括菜单、工具栏和模型编辑区。下面依次介绍各部分的组成。

菜单中包括了“File”、“Edit”、“View”、“Simulation”、“Format”、“Tools”和“Help”。其中，(1)“File”子菜单主要用于模型文件的打开、保存、属性、打印等设置，如图 11.5 所示，其中，

- “Save”菜单项：用于保存 Simulink 的系统模型，模型文件的存储形式为后缀为“mdl”的模型文件，模型文件中的模块框图和各模块的相关属性将被保存下来。
- “Save As”菜单项：用于对已有的模型执行另存为操作。
- “Model Properties”菜单项：模型属性信息的描述。
- “Preferences”菜单项：Simulink 模型编辑界面设置。
- “Print...”菜单项：打印模型。
- “Print Details”菜单项：打印模型的细节设置。

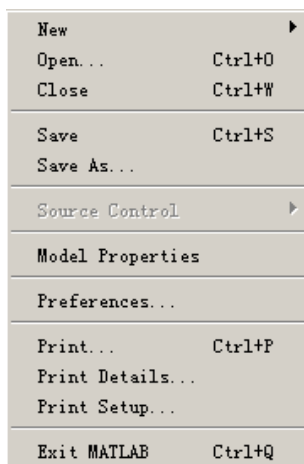


图 11.5 Simulink 模型编辑窗口的“File”子菜单

(2)“Edit”子菜单主要用于模型设计过程中的常用复制、粘贴等操作，如图 11.6 所示，其中，

- “Paste Duplicate Inport”菜单项：用于粘贴、复制的输入模块。
- “Copy Model To Clipboard”菜单项：复制模型到剪贴板。
- “Create Subsystem”菜单项：创建子系统模块。
- “Mask Subsystem...”菜单项：封装子系统模块。
- “Look Under Mask”菜单项：查看封装的子系统的内部结构。
- “Refresh Model Blocks”菜单项：刷新模型输入/输出和参数的设置。
- “Update Diagram”菜单项：更新模型框图的外观。

Undo Add Line	Ctrl+Z
Redo Move	Ctrl+Y
Cut	Ctrl+X
Copy	Ctrl+C
Paste	Ctrl+V
Paste Duplicate Import	
Delete	Delete
Select All	Ctrl+A
Copy Model To Clipboard	
Find...	Ctrl+F
Open Block	
Explore	
Block Properties...	
Create Subsystem	Ctrl+G
Mask Subsystem...	Ctrl+M
Look Under Mask	Ctrl+U
Link Options	
Refresh Model Blocks	Ctrl+K
Update Diagram	Ctrl+D

图 11.6 Simulink 模型编辑窗口的“Edit”子菜单

(3) “View”子菜单主要用于窗口显示的控制，如图 11.7 所示，其中，

- “Go To Parent”菜单项：用于显示当前模型系统的父系统。
- “Toolbar”菜单项：控制工具条的显示。
- “Status Bar”菜单项：控制状态栏的显示。
- “Model Browser Options”菜单项：显示模型浏览器的设置，其中 Model Browser 子菜单可在模型编辑窗口的左侧显示模型的树形文件浏览界面。
- “Block Data Tips Options”菜单项：当鼠标位于模块上方时，显示模块内部的数据。
- “System Requirements”菜单项：系统要求设置。
- “Library Browser”菜单项：显示 Simulink 模型库浏览器。
- “Model Explorer”菜单项：显示模型浏览器，可显示 Simulink 模块库中的模块和用户建立模型的属性和内容。
- “Zoom In”菜单项：对模型执行缩小操作。
- “Zoom Out”菜单项：对模型执行放大操作。
- “Fit Selection To View”菜单项：自动选择最合适的比例在窗口中显示模型。
- “Normal”菜单项：以正常比例（100%）显示模型。

Go To Parent
✓ Toolbar
✓ Status Bar
Model Browser Options ▶
Block Data Tips Options ▶
System Requirements ▶
Library Browser
Model Explorer
Zoom In
Zoom Out
Fit Selection To View
Normal (100%)
Port Values ▶
Remove Highlighting
Highlight ... ▶

图 11.7 Simulink 模型编辑窗口的“View”子菜单

(4) “Simulation” 子菜单主要用于控制仿真，如图 11.8 所示。其中，

- “Start” 菜单项：启动 Simulink 模型仿真。
- “Stop” 菜单项：停止 Simulink 模型仿真。
- “Configuration Parameters...” 菜单项：设置 Simulink 模型仿真参数。
- “Normal” 菜单项：普通 Simulink 模型，默认选项。
- “Accelerator” 菜单项：加速 Simulink 模型。
- “External” 菜单项：扩展 Simulink 模型。

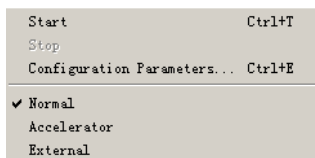


图 11.8 Simulink 模型编辑窗口的“Simulation”子菜单

(5) “Format” 子菜单主要用于对模块的相关格式设置，如图 11.9 所示。其中：

- “Font” 菜单项：文字的字体设置。
- “Text Alignment” 菜单项：标注文字对齐方式设置。
- “Enable TeX Commands” 菜单项：使 Tex 控制命令有效。
- “Flip Name” 菜单项：翻转模块名。
- “Flip Block” 菜单项：翻转模块。
- “Rotate Block” 菜单项：旋转模块。
- “Hide Name” 菜单项：隐藏/显示模块的名字。
- “Show Drop Shadow” 菜单项：显示/隐藏模块的阴影效果。
- “Foreground Color” 菜单项：设置模块的前景色。
- “Background Color” 菜单项：设置模块的背景色。
- “Screen Color” 菜单项：设置屏幕颜色。
- “Port/Signal Displays” 菜单项：控制端口/信号显示。
- “Block Displays” 菜单项：控制模块显示。
- “Library Link Display” 菜单项：控制库链接显示。



图 11.9 Simulink 模型编辑窗口的“Format”子菜单










(6) “Tools” 子菜单主要用于打开 Simulink 建模仿真的工具，如图 11.10 所示。其中，

- “Simulink Debugger...” 菜单项：打开 Simulink 调试器。
- “Fixed-Point Settings” 菜单项：对 Simulink 仿真结果的浮点设置。

- “Model Advisor” 菜单项：打开模型咨询工具。
- “Lookup Table Editor” 菜单项：打开查表编辑工具。
- “Data Class Designer” 菜单项：打开用户定义数据类型工具。
- “Bus Editor” 菜单项：打开线路编辑工具。
- “Signal & Scope Manager” 菜单项：信号和示波器管理器工具。
- “Real-Time Workshop” 菜单项：打开实时工作平台。
- “External Mode Control Panel” 菜单项：打开外部的模式控制平台。
- “Control Design” 菜单项：控制设计工具。
- “Parameter Estimation” 菜单项：参数估计工具。
- “Report Generator” 菜单项：仿真报告生成器。



图 11.10 Simulink 模型编辑窗口的“Tools”子菜单

工具栏包括与其他窗口工具栏类似的文件常用的新建、打开、复制、打印、剪切等操作工具，同时还包括专用于模型文件操作的工具 （开始仿真）、（停止仿真）、（字体设置）、（鼠标悬停在模块上方时显示）、（创建并编译生成 exe 文件）、（刷新模型输入/输出和参数的设置）、（更新模型框图）、（创建子系统）、（打开 Simulink 模型库浏览窗口）。

模型编辑区用于设计 Simulink 仿真模型，主要通过简单的鼠标单击和拖曳操作实现，在后面的章节中将会详细介绍建模仿真的具体实现。

11.2 Simulink 常用基本模块

在前面的章节中已介绍了 MATLAB 在“Simulink Library Browser”窗口中提供了大量可用于仿真建模的模块，使用这些现成的模块将有助于我们快速、方便地设计出仿真模型。Simulink 常用基本模块包括公共模块和专业模块。

为了便于用户更好地使用 Simulink 内置模块库，本节将对 Simulink 公共模块库中的各模块的功能做简单的介绍，而其他的主要应用于专业领域的专业模块库，一般用户可能较少涉及，本书不详细展开叙述。

Simulink 公共模块库共涉及 16 个子模块库，如图 11.11 所示。

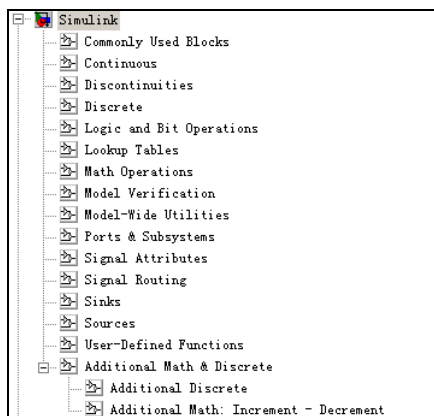


图 11.11 Simulink 的 16 个公共模块库

1. Commonly Used Blocks（常用模块库）

Simulink 为便于用户使用，从其他的模块库中将常用的模块组合在一起构建了常用模块库，该模块库总计包含了 22 个模块。单击 Commonly Used Blocks 模块库，在“Simulink Library Browser”窗口的右侧将显示如图 11.12 所示的常用模块，模块的具体功能将在后续各模块库的讲解中介绍。

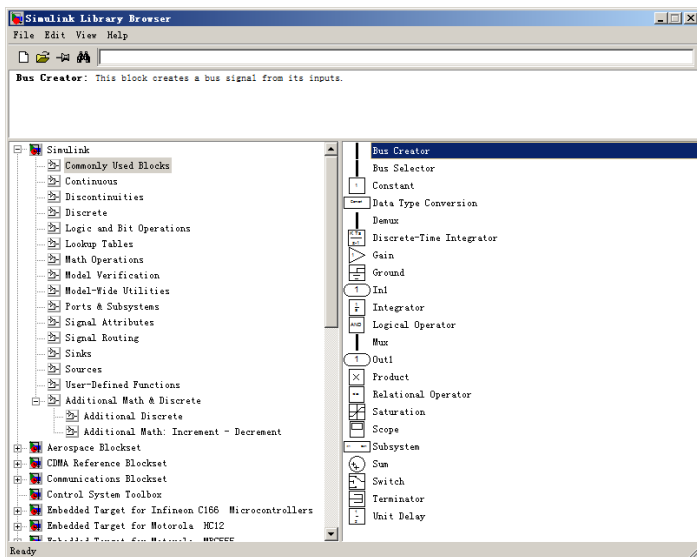


图 11.12 Simulink 的常用模块库

2. Continuous（连续模块库）

连续模块库为仿真提供连续模块系统。各模块的图标和名称如图 11.13 所示，其中，

- Derivative 模块：连续信号的微分运算。
- Integrator 模块：连续信号的积分运算。
- State-Space 模块：实现线性状态空间系统。
- Transfer Fcn 模块：实现线性的传递函数。
- Transport Delay 模块：对输入信号进行传输延迟。
- Variable Transport Delay 模块：对输入信号进行可变时间的传输延迟。
- Zero-Pole 模块：实现零极点形式的传递函数。

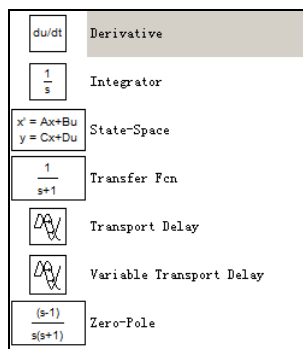


图 11.13 Simulink 的连续模块库

3. Discontinuities (非连续模块库)

非连续模块库为仿真提供非连续状态的模块。各模块的图标和名称如图 11.14 所示，其中，

- Backlash 模块：实现输入/输出相同变化的模型。
- Coulomb & Viscous Friction 模块：模拟原点不连续，其余点为线性的库仑力和黏滞力模块。
- Dead Zone 模块：死区模块，控制输入信号在指定范围。
- Dead Zone Dynamic 模块：动态死区模块，根据输入信号的范围控制输入信号在指定范围。
- Hit Crossing 模块：当输入信号的值大于 Hit Crossing 参数值，模块输出为 1，否则输出为 0。
- Quantizer 模块：对输入信号进行量化处理的模块。
- Rate Limiter 模块：限速模块，通过模块信号的一阶导数限制信号的变化。
- Rate Limiter Dynamic 模块：动态限速模块，限制信号的递增和递减速率。
- Relay 模块：实现继电器功能，控制输入参数的下降。
- Saturation 模块：饱和度模块，用于限制输入信号的上下限。
- Saturation Dynamic 模块：动态饱和度模块。
- Wrap To Zero 模块：限零模块，用于控制模块输出为零信号。

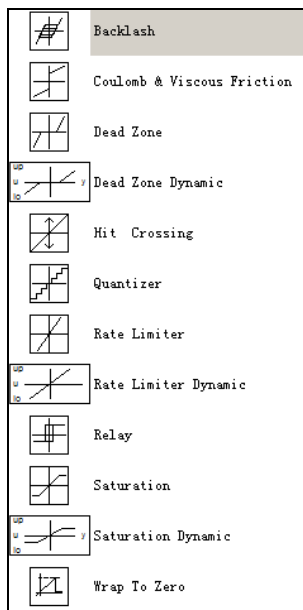


图 11.14 Simulink 的非连续模块库

4. Discrete（离散系统模块库）

离散系统模块库为仿真提供离散元件。各模块的图标和名称如图 11.15 所示，其中，

- Difference 模块：差分模块。
- Discrete Derivative 模块：离散微分模块。
- Discrete Filter 模块：离散过滤分析模块，实现无限脉冲响应（IIR）和有限脉冲响应（FIR）的滤波器。
- Discrete State-Space 模块：离散状态空间模块。
- Discrete Transfer Fcn 模块：实现离散传递函数。
- Discrete Zero-Pole 模块：实现用零极点表达的离散传递函数。
- Discrete-Time Integrator 模块：离散时间积分模块。
- First-Order Hold 模块：模块用于实现一阶采样保持。
- Integer Delay 模块：延迟信号 N 个采样周期。
- Memory 模块：保持输入和输出不变。
- Tapped Delay 模块：延迟标量信号多个采样周期，并输出所有版本。
- Transfer Fcn First Order 模块：实现离散系统的一阶传递函数。
- Transfer Fcn Lead or Lag 模块：实现离散系统超前或滞后的传递函数。
- Transfer Fcn Real Zero 模块：实现离散系统的带零点的传递函数。
- Unit Delay 模块：单位采样时间延迟模块。
- Weighted Moving Average 模块：实现加权平均模块。
- Zero-Order Hold 模块：实现采样的零阶保持，即在一个采样周期内输出信号保持不变。

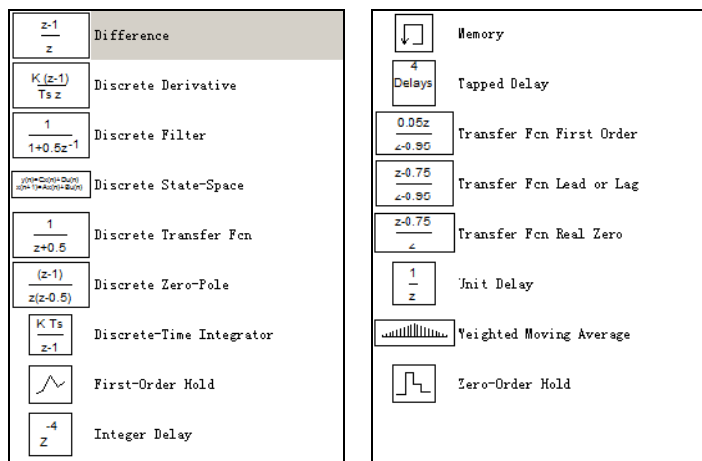


图 11.15 Simulink 的离散系统模块库

5. Logic and Bit Operations（逻辑和位运算模块库）

逻辑和位运算模块库用于提供逻辑运算和位运算的模块。各模块的图标和名称如图 11.16 所示，其中，

- Bit Clear 模块：实现位清零操作。
- Bit Set 模块：实现位置位操作。
- Bitwise Operator 模块：实现位逻辑运算操作。
- Combinatorial Logic 模块：实现真值表功能。
- Compare To Constant 模块：用于信号常量比较。

- Compare To Zero 模块：用于信号和零比较。
- Detect Change 模块：用于检测信号的变化。
- Detect Decrease 模块：用于检测信号的递减。
- Detect Fall Negative 模块：用于检测信号变负。
- Detect Fall Nonpositive 模块：用于检测信号非正。
- Detect Increase 模块：用于检测信号递增。
- Detect Rise Nonnegative 模块：用于检测信号非负。
- Detect Rise Positive 模块：用于检测信号变正。
- Extract Bits 模块：实现提取位操作。
- Interval Test 模块：检测信号是否在指定的范围。
- Interval Test Dynamic 模块：动态检测信号范围。
- Logical Operator 模块：实现逻辑运算操作。
- Relational Operator 模块：实现关系运算操作。
- Shift Arithmetic 模块：实现移位运算。

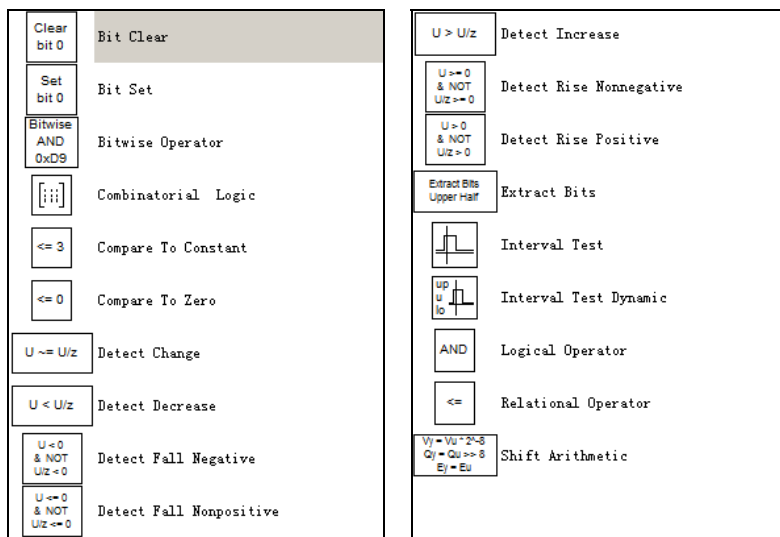


图 11.16 Simulink 的逻辑和位运算模块库

6. Lookup Tables (查表模块库)

查表模块库主要用于对信号的查表和插值运算。各模块的图标和名称如图 11.17 所示，其中，

- Cosine 模块：通过查找表法实现余弦函数。
- Direct Lookup Table (n-D) 模块：用于表数据选择。
- Look up Table 模块：实现输入信号的一维线性内插。
- Look up Table(2-D)模块：实现输入信号的二维线性内插。
- Look up Table(n-D)模块：实现输入信号的 n 维线性内插。
- PreLookup Index Search 模块：查找输入信号的位置。
- Sine 模块：通过查找表法实现正弦函数。

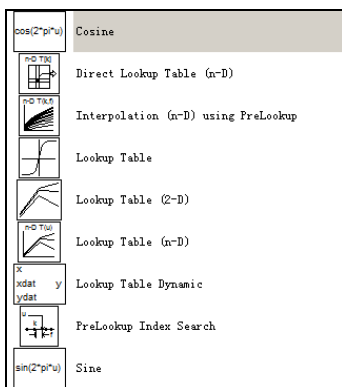


图 11.17 Simulink 的查表模块库

7. Math Operations（数学运算模块库）

数学运算模块库提供数学运算功能模块。各模块的图标和名称如图 11.18 所示，其中，

- Abs 模块：计算信号的绝对值。
- Add 模块：对输入信号实现相加或相减操作。
- Algebraic Constraint 模块：限制输入信号为零。
- Assignment 模块：用于对输入信号赋值。
- Bias 模块：用于对输入信号加上偏差。
- Complex to Magnitude–Angle 模块：用于双精度信号转换为幅值和幅角。
- Complex to Real–Imag 模块：用于双精度信号转换为复数信号的实部和虚部。
- Divide 模块：用于输入信号的乘除运算。
- Dot Product 模块：用于输入信号的点乘。
- Gain 模块：将输入信号乘上一个增益。
- Magnitude–Angle to Complex 模块：将幅值和幅角的信号转换成为复数信号。
- Math Function 模块：实现多种数学函数操作。
- Matrix Concatenation 模块：进行水平或垂直的串联。
- MinMax 模块：计算输入信号的最小值和最大值。
- MinMax Running Resettable 模块：计算输入信号可复位的最小值和最大值。
- Polynomial 模块：计算输入信号的多项式值。
- Product 模块：对输入信号进行乘除运算。
- Real–Imag to Complex 模块：用于将信号的实部和虚部转换成为复数信号。
- Reshape 模块：实现输入信号的维数转换。
- Rounding Function 模块：实现输入信号的四舍五入取整运算。
- Sign 模块：用于实现信号的符号函数操作。
- Sine Wave Function 模块：输出正弦函数模块。
- Slider Gain 模块：滑块增益模块。
- Subtract 模块：实现输入信号的加减运算。
- Sum 模块：输入信号的求和模块。
- Sum of Elements 模块：实现输入信号的元素求和模块。
- Trigonometric Function 模块：实现三角函数运算。
- Unary Minus 模块：用于信号求负值。

- Weighted Sample Time Math 模块：对加权采样的输入信号进行加、减、乘、除运算。

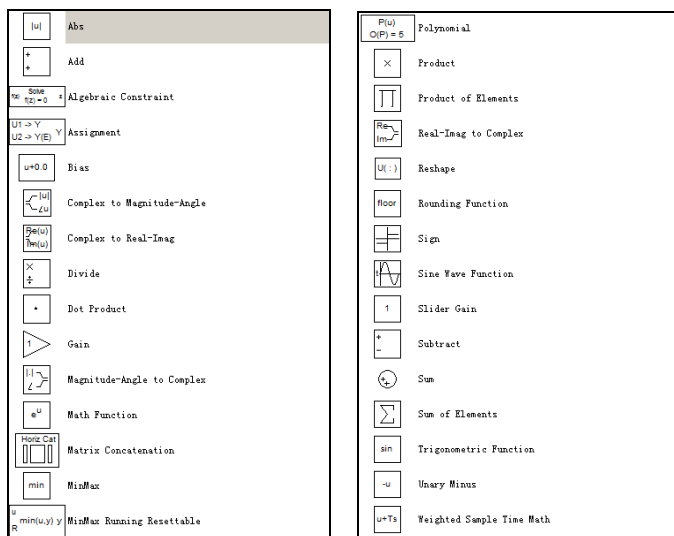


图 11.18 Simulink 的数学运算模块库

8. Model Verification (模型验证模块库)

模型验证模块库提供了对信号的检测模块。各模块的图标和名称如图 11.19 所示，其中，

- Assertion 模块：确定输入信号为非零信号。
- Check Discrete Gradient 模块：确定离散采样的离散信号的差分绝对值是否小于上限。
- Check Dynamic Gap 模块：检查输入信号的动态间隙。
- Check Dynamic Lower Bound 模块：确定一个信号是否总是小于另一个信号。
- Check Dynamic Range 模块：确定一个信号是否总是在另两个信号之间。
- Check Dynamic Upper Bound 模块：确定一个信号是否总是大于另一个信号。
- Check Input Resolution 模块：检查输入信号的分辨率。
- Check Static Gap 模块：检查输入信号的静态间隙。
- Check Static Lower Bound 模块：确定输入信号是否小于静态下限。
- Check Static Range 模块：检查输入信号的静态范围。
- Check Static Upper Bound 模块：检查输入信号是否大于静态上限。

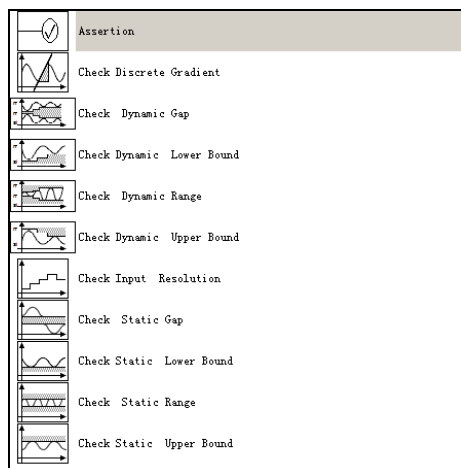


图 11.19 Simulink 的模型验证模块库

9. Model-Wide Utilities（模型扩充实用模块库）

模型扩充实用模块库提供进行模型扩充的实用模块库。各模块的图标和名称如图 11.20 所示，其中，

- Block Support Table 模块：支持表模块。
- DocBlock 模块：保存模型描述信息模块。
- Model Info 模块：模型控制信息。
- Timed-Based Linearization 模块：在指定时间生成线性模型。
- Trigger-Based Linearization 模块：触发时在基本空间产生线性模型。

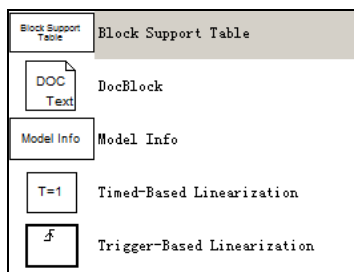


图 11.20 Simulink 的模型扩充实用模块库

10. Ports & Subsystems（端口和子系统模块库）

端口和子系统模块库用于控制端口和子系统的执行。各模块的图标和名称如图 11.18 所示，其中，

- Configurable Subsystem 模块：可配置子系统模块。
- Atomic Subsystem 模块：原子系统模块。
- CodeReuseSubsystem 模块：代码重用子系统模块。
- Enable 模块：启动系统模块。
- Enabled and Triggered Subsystem 模块：启动触发子系统模块。
- Enabled Subsystem 模块：启动子系统模块。
- For Iterator Subsystem 模块：For 循环子系统模块。
- Function-Call Generator 模块：函数调用产生模块。
- Function-Call Subsystem 模块：函数调用子系统模块。
- If 模块：If 条件模块。
- If Action Subsystem 模块：If 条件子系统模块。
- In1 模块：子系统的输入端口模块。
- Model 模块：模型模块。
- Out1 模块：子系统的输出端口模块。
- Subsystem 模块：包含输入和输出的通用子系统。
- Subsystem Examples 模块：子系统举例模块。
- Switch Case 模块：输入端口执行选择操作模块。
- Switch Case Action Subsystem 模块：输入端口执行选择操作子系统模块。
- Trigger 模块：子系统的触发端口。
- Triggered Subsystem 模块：触发子系统模块。
- While Iterator Subsystem 模块：While 循环执行子系统。

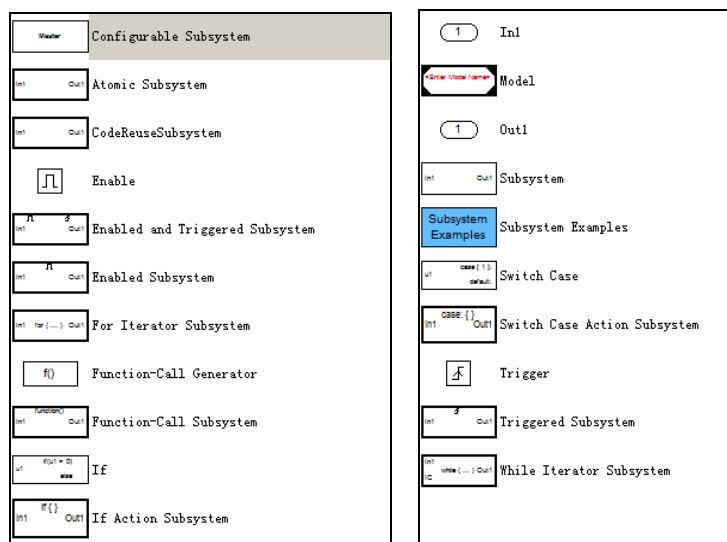


图 11.21 Simulink 的端口和子系统模块库

11. Signals Attributes (信号属性模块库)

信号属性模块库用于对信号的类型等进行设置。各模块的图标和名称如图 11.22 所示，其中，

- Data Type Conversion 模块：信号类型转换模块。
- Data Type Conversion Inherited 模块：信号类型转换模块，转换第二个输入数据的数据类型。
- Data Type Duplicate 模块：强制所有输入信号为相同的类型。
- Data Type Propagation 模块：设置数据类型和参照指定的信号归一化传播信号。
- Data Type Propagation Examples 模块：为数据类型模块举例。
- Data Type Scaling Strip 模块：数据类型剔除模块。
- IC 模块：初始条件设置模块。
- Probe 模块：信号探测器模块。
- Rate Transition 模块：速率转换模块。
- Signal Conversion 模块：信号转换模块。
- Signal Specification 模块：指定模块的信号线属性。
- Weighted Sample Time 模块：以加权采样时间对输入信号进行加、减、乘、除运算。
- Width 模块：检测信号宽度模块。

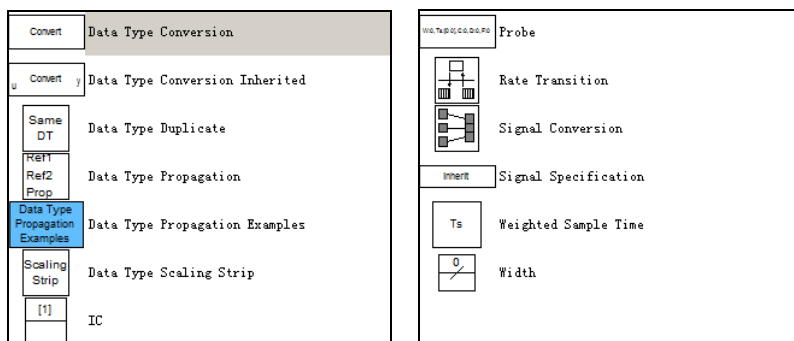


图 11.22 Simulink 的信号属性模块库

12. Signal Routing（信号路由模块库）

信号路由模块库用于控制输入、输出信号的相关处理。各模块的图标和名称如图 11.22 所示，其中，

- Bus Assignment 模块：对总线信号分解。
- Bus Creator 模块：根据输入信号产生总线信号。
- Bus Selector 模块：用于选择的总线信号。
- Data Store Memory 模块：定义共享数据存储空间。
- Data Store Read 模块：从共享数据存储空间读出数据。
- Data Store Write 模块：向共享数据存储空间写入数据。
- Demux 模块：输出分解的信号。
- Environment Controller 模块：模块用于控制环境。
- From 模块：从 Goto 模块中获得信号。
- Goto 模块：向 Goto 模块传递信号。
- Goto Tag Visibility 模块：控制 Goto 模块标记的可视化。
- Index Vector 模块：用于索引向量。
- Manual Switch 模块：手动选择。
- Merge 模块：合并输入信号。
- Multiport Switch 模块：用于端口选择。
- Mux 模块：输入信号的组合。
- Selector 模块：信号选择器。
- Switch 模块：输出选择器。

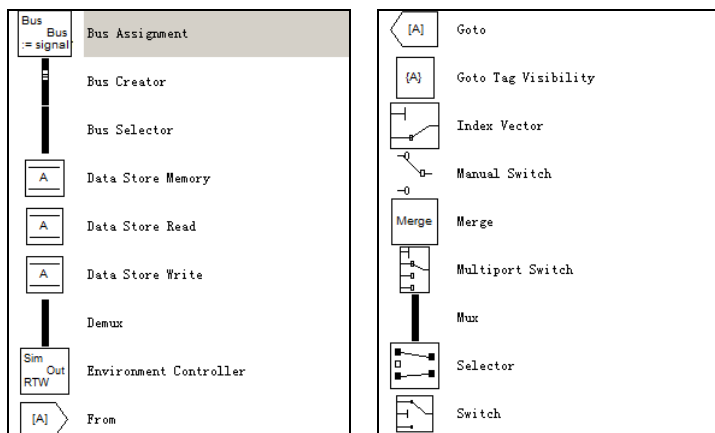


图 11.23 Simulink 的信号路由模块库

13. Sinks（输出模块库）

输出模块库为仿真提供输出设备模块。各模块的图标和名称如图 11.24 所示，其中，

- Display 模块：显示输入信号的值。
- Floating Scope 模块：用于显示浮点信号。
- Out1 模块：提供子系统或模型输出端口。
- Scope 模块：示波器模块，用于显示输入信号与仿真时间的关系曲线。
- Stop Simulation 模块：当输入信号为 0 时停止仿真。
- Terminator 模块：实现输出信号的终止。

- To File 模块：将仿真数据写入文件。
- To Workspace 模块：将仿真数据写入 MATLAB 工作空间。
- XY Graph 模块：在图形窗口显示信号的二维图。

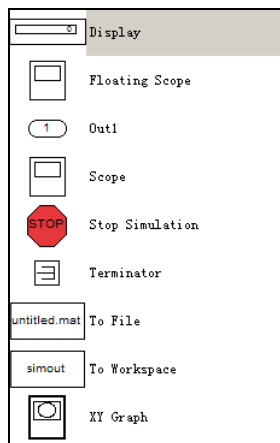


图 11.24 Simulink 的输出模块库

14. Sources（输入模块库）

输入模块库主要用于为仿真系统提供各种输入信号源。各模块的图标和名称如图 11.25 所示，其中，

- Band-Limited White Noise 模块：限制带宽白噪声，产生正态分布的随机数。
- Chirp Signal 模块：输出一个正弦波信号，其频率随时间线性增加。
- Clock 模块：显示当前的仿真时间。
- Constant 模块：生成常数值信号。
- Counter Free-Running 模块：模块为自由运行的计数器。
- Counter Limited 模块：模块为受限的计数器。
- Digital Clock 模块：数字时钟显示仿真时间。
- From File 模块：从文件读取获取输入信号。
- From Workspace 模块：从工作空间内读入输入信号。
- Ground 模块：输入信号为接地模块。
- In1 模块：提供一个子系统或模型的输入端口。
- Pulse Generator 模块：产生脉冲信号。
- Ramp 模块：产生斜坡信号。
- Random Number 模块：产生正态分布的随机信号。
- Repeating Sequence 模块：产生周期信号。
- Repeating Sequence Interpolated 模块：产生内插的周期信号。
- Repeating Sequence Stair 模块：产生梯阶的周期信号。
- Signal Builder 模块：信号生成模块。
- Signal Generator 模块：产生不同的信号，包括正弦波、方波、锯齿波。
- Sine Wave 模块：生成正弦波信号。
- Step 模块：生成阶跃信号。
- Uniform Random Number 模块：生成均匀分布的随机信号。

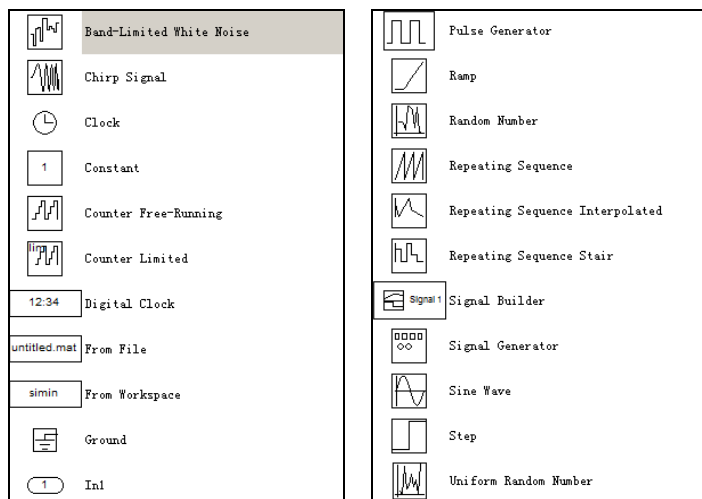


图 11.25 Simulink 的输入模块库

15. User-defined Functions（用户自定义模块库）

用户自定义模块库用于通过 S 函数定义自设的模块，封装好的模块在下次使用的时候可以直接调用。各模块的图标和名称如图 11.26 所示。这里不再详细展开，感兴趣的读者可以在运用中阅读模块相应的描述信息。

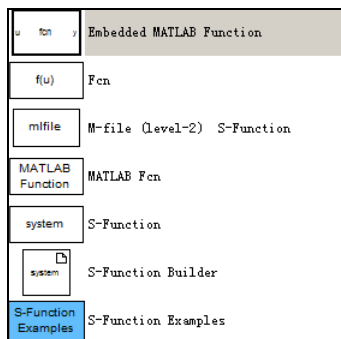


图 11.26 Simulink 的用户自定义模块库

16. Additional Math & Discrete（其他数学和离散模块库）

其他数学和离散模块库提供了其他的数学模块和离散模块，如图 11.27 和 11.28 所示为各模块的图标和名称。对于一般的用户可能不是很常用，这里不详细叙述。

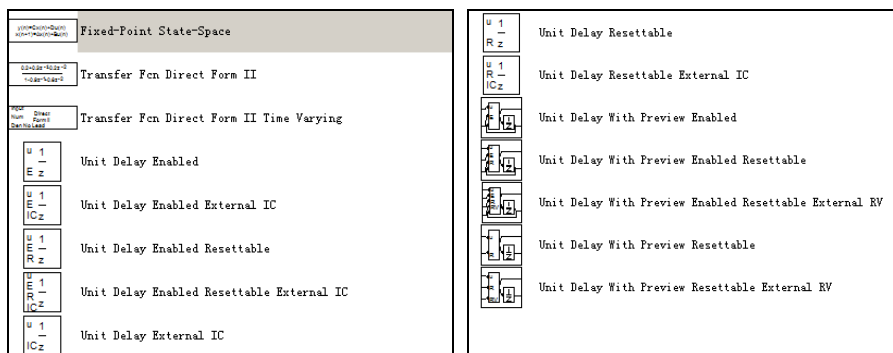


图 11.27 Simulink 的其他离散模块库

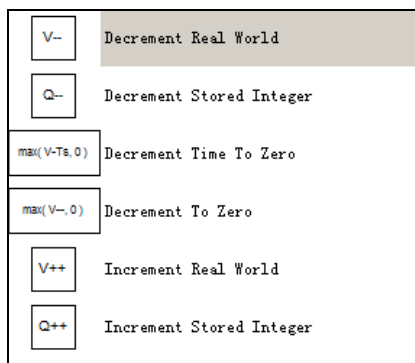


图 11.28 Simulink 的其他数学模块库

11.3 Simulink 建模与仿真

通过上述章节的学习相信读者对 Simulink 的建模编辑窗口和基本的模块库已有了初步的了解。在本节中向用户介绍如何进行 Simulink 建模与仿真，将按建模的基本流程来介绍。最后，以一简单的实例演示 Simulink 建模与仿真的过程。

利用 Simulink 进行建模仿真的一般过程如下。

- (1) 新建模型编辑窗口。
- (2) 从模块库中选择需要的模块放到模型编辑窗口。
- (3) 执行模块基本操作，包括设置模块名，调整模块大小等基本操作。
- (4) 连接模块间的信号线。
- (5) 设置模块参数。
- (6) 仿真运行，若出错后调试仿真程序。
- (7) 保存仿真结果。

11.3.1 选择模块

在模块浏览窗口根据建模仿真的任务选择可以实现指定功能的模块，并添加到仿真编辑窗口。各模块的功能在前面的章节中已简单地介绍过了，本节将主要介绍如何方便地详细了解你需要的模块并放置到模型中。

1. 模块的详细功能信息的获取

用户首先应该根据建模需要选择模块，可以根据各模块库功能的关键字在查找对话框中查找所需模块，或者根据模块的功能在不同的模块库中直接选择需要的模块。

对各模块功能的了解主要可以通过下面两种方式。

- (1) 选中相应的模块，在模块浏览窗口的上方将显示该模块的描述信息，如图 11.29 所示。
- (2) 选中相应的模块并单击鼠标右键，在弹出的快捷菜单中选择“Help for+指定模块”，即可打开相应的模块的帮助主题，如图 11.30 所示，查找模块“Bus Selector”的帮助信息。当然也可以在帮助浏览窗口中直接输入查找指定模块的帮助信息。

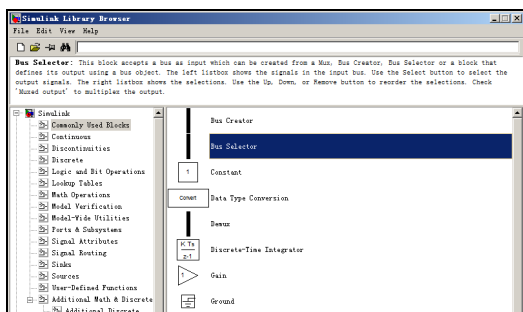


图 11.29 Simulink 模块描述信息的获取

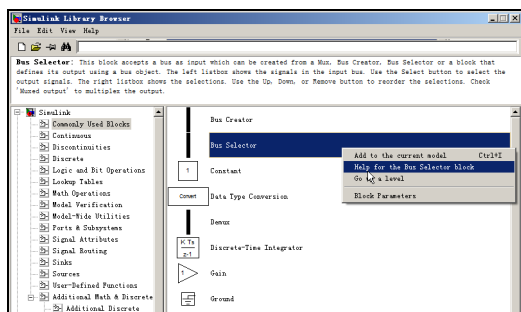


图 11.30 Simulink 模块帮助信息的获取

2. 模块的添加

在查找到指定功能的模块后，即可把模块添加到仿真模型中。模块添加的方法有以下 3 种。

- 鼠标拖曳法：用鼠标左键按住待添加的模块不放，拖至模型编辑窗口再松开。
- 下拉式菜单法：选中待添加的模块，选择“Edit”→“Add to the current model”命令，指定模块即可添加到当前模型中。
- 弹出式菜单法：对待添加的模块执行右键操作，选择“Add to the current model”命令。

11.3.2 模块基本操作

在模型编辑窗口中添加了模块后，界面往往并不能满足最后模型的需要，可以对模块执行一些基本操作。这些基本操作包括模块命名、模块移动、剪切、复制、粘贴等基本操作，下面具体介绍如何实现这些操作。

1. 模块移动、剪切、复制、粘贴

模块移动很简单，同一模型编辑窗口内，直接使用鼠标左键拖曳到相应位置即可；在不同模型编辑窗口之间移动模块，需在移动的时候同时按下“Shift”键。

模块的剪切、复制、粘贴也与 Office 的一般操作类似，同时模型编辑窗口也提供了执行相应操作的菜单。

2. 模块大小的调整

模块大小的调整也很简单，可以通过鼠标直接拉伸或缩放模块外的四边的顶点，即可完成模块大小的调整。

3. 模块方向调整

选定指定模块，选择“Format”→“Rotate Block”命令可使模块旋转 90°，选择“Format”→“Flip Block”命令可使模块旋转 180°。

4. 模块名

模块名的操作包括模块名的显示、修改、位置调整。下面具体介绍其实现方法。

1) 模块名的显示

“Format”菜单下的 Hide Name 和 Show Name 分别控制模块名的隐藏与显示。

2) 模块名的修改

单击模块名需要修改的地方，模块名即进入修订状态，可直接修改模块名。模块名的字体及其大小也可以通过 Font 菜单项修改。在对模块名修改的时候需要注意不能包含中文及西腊字母。

3) 模块名的位置调整

模块名的位置：一般当模块的接口为上下时，模块名只能位于模块的左右位置；当模块的接

口为左右时，模块名只能位于模块的上下位置。选择“Format”→“Flip Name”命令，可实现模块名位置的移动，即上下和左右移动。

5. 模块颜色设置

选定模块，模块颜色主要是通过“Format”菜单来实现，其中，

- Foreground color: 设置模块的前景色。
- Background color: 设置模块的后景色。
- Screen color: 设置模块的颜色。

6. 模块阴影的添加

选定模块，选择“Format”→“Show Drop Shadow”命令可使模块产生阴影效果。

7. 模块属性设置

选中指定的模块，选择“Edit”→“Block Properties”命令可以对模块属性设置，包括的属性有 Description（模块的描述性信息）、Priority（模块的优先级属性）、Tag（模块的标签属性）、Open function（设置模块的回调函数）和 Attributes format string（属性格式字符串）属性。

11.3.3 信号线操作

信号线主要用于连接不同的模块，将一个模块的输出传递为另一个模块的输入。本节将主要介绍信号线的基本操作，包括如何利用信号线实现模块的连接，并在连接后设置信号线的粗细、分支、标签等。

1. 模块的连接

信号线可实现模型信号的传递，在仿真的时候信号可通过信号线从一个模块传递到另一个模块。模块连接的操作比较简单，用户只需将鼠标指针移动到模块的输出端口，在出现“+”符号的时候，用鼠标拖动到另一个需要连接的模块即可。连接后两模块将出现黑色箭头的连线，箭头指向输出模块，同时如果两个模块的连接为无效连接将出现红色的虚线，如图 11.31 所示，左边为正确的连接，右边为错误的连接。



图 11.31 Simulink 模块的连接

2. 信号线的分支

根据仿真建模的实际需要，有时需要对信号线分支，连接不同的输出。信号线的分支通过鼠标右键实现，在信号线的分支点按住鼠标右键，拖出分支线，如图 11.32 所示。

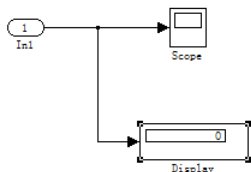


图 11.32 Simulink 信号线的分支

3. 信号线的折弯

有时对于一些比较复杂的模型，涉及较多的模块，如果仅通过直的横线或竖线连接模块，可能无法回避中间的模块。信号线的折弯操作可以改变连线的走向，从而绕开中间的模块。可以通过鼠标直接拉出折弯的信号线，同时按下“Shift”键的同时，信号线的折弯可以产生斜线的效果，如图 11.33 所示。

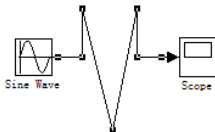


图 11.33 Simulink 信号线的折弯

4. 信号线的标记

为了便于用户更好地理解模型，可以在模块连接的信号线上做标记。添加信号线标记的方法很简单，只要双击信号线上需要添加标记的位置，即可出现输入标记的文本框，直接输入需要添加的标记，如图 11-34 所示。

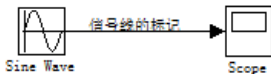


图 11.34 Simulink 信号线的标记

5. 信号线的粗细

信号线的粗细是由信号线引出的信号的类型决定的，但选择“Format”→“Port Signal Display”→“Wide Vector Lines”命令时，如果为标量信号，则为细线，如果为向量信号，则为粗线。选择“Format”→“Port Signal Display”→“Vector Line Widths”命令时则可以显示向量信号线的宽度，即向量信号由多少个单一信号组成。

6. 信号线的分离

如果在设计中发生了连接错误，需要把模块与信号线分离，可以按住“Shift”键再拖动模块即可。

11.3.4 参数设置

当连接好模块后，在开始仿真前还需要设置模块和仿真的相应参数，才可以完成仿真功能。

1. 模块参数

不同模块设置的参数也不同，用鼠标双击模块即可打开模块参数设置对话框。以 Sine Wave 输入模块为例介绍模块参数的设置。

【例 11.1】 Sine Wave 输入模块参数的设置。

Sine Wave 输入模块主要用于显示仿真的结果，双击 Sine Wave 输入模块将打开如图 11.35 所示的 Sine Wave 输入模块设置对话框。在模块设置对话框中可以根据实际的需要设置相应的参数，例如本例中包含的参数有 Sine type（sine 函数的类型）、Time（模拟的横坐标）、Amplitude（幅值）、Bias（偏差）、Frequency（频率）、Phase（相位）和 Sample Time（采用周期）。

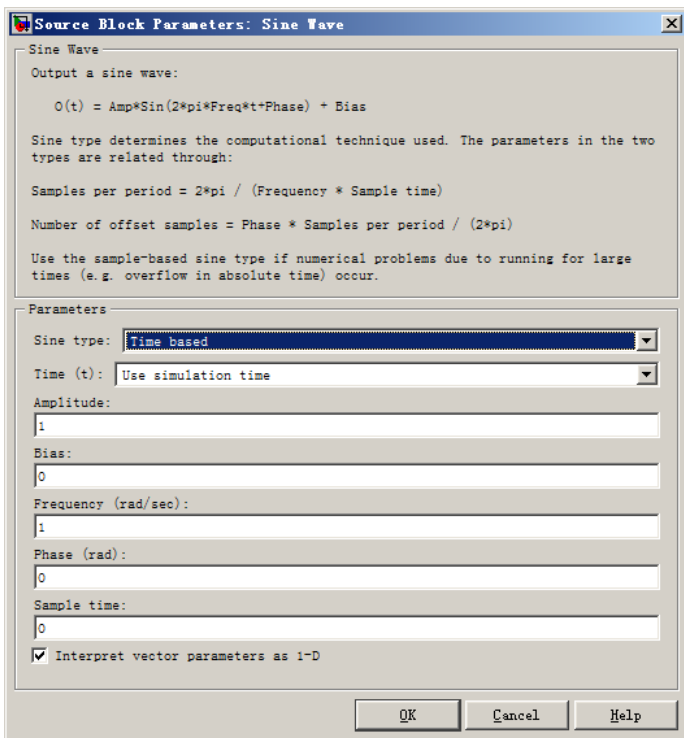


图 11.35 Simulink Sine Wave 输入模块参数的设置

2. 仿真参数

本节将介绍仿真参数的设置，选择“Simulation”→“Configuration Parameters”命令，将打开如图 11.36 所示的 Simulink 仿真参数的设置对话框。Simulink 可以设置的仿真参数有 Solver（算法）、Data Import/Export（数据输入/输出）、Optimization（优化）、Diagnostics（诊断）、Hardware Implementation（硬件工具）、Model Referencing（模块引用）和 Real-Time Workshop（实时工作）。

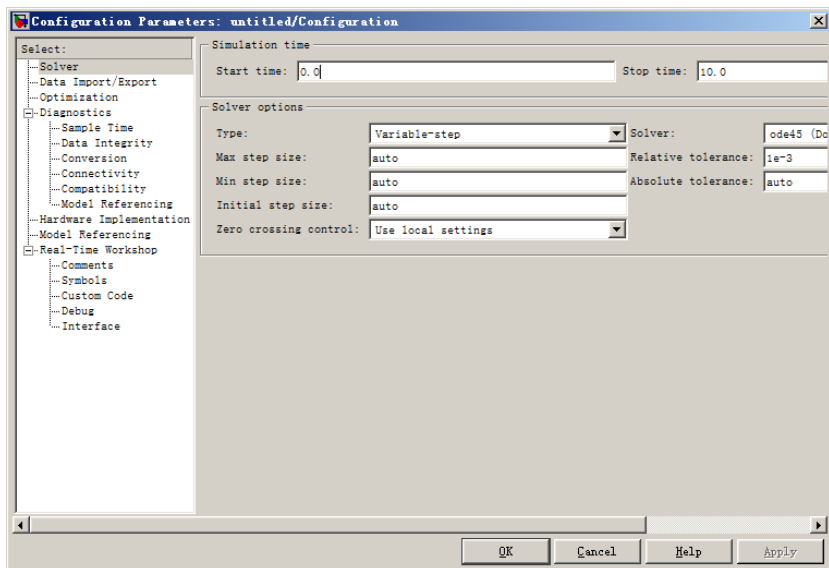


图 11.36 Simulink 仿真参数的设置

下面具体介绍各仿真参数的设置。

1) Solver 参数

算法参数主要用于设置仿真过程的一些基本参数,如图 11.37 所示,包括 Simulation time 和 Solver options 两个区域。其中,

- Simulation time: 用于设置仿真的时间,包括 Start time (起始时间) 和 Stop time (结束时间)。
- Solver options: Type (仿真类型) 包括 Variable-step (变步长) 和 Fixed-step (固定步长); Solver (仿真解的算法), 当设置为变步长时, 仿真算法有 ode45、ode23、ode113、ode15s、ode23s、ode23t、ode23tb 和 discrete, 当设置为固定步长时, 仿真算法有 ode5、ode4、ode3、ode2、ode1 和 discrete; 根据选择的仿真类型和仿真算法将显示具体算法参数的设置, 对于变步长类型, 可以设置 Max/Min step size (步长参数范围) 和 Initial step size (初始步长参数), 同时可以设置仿真精度, 包括 Relative tolerance (相对误差) 和 Absolute tolerance (绝对误差)。

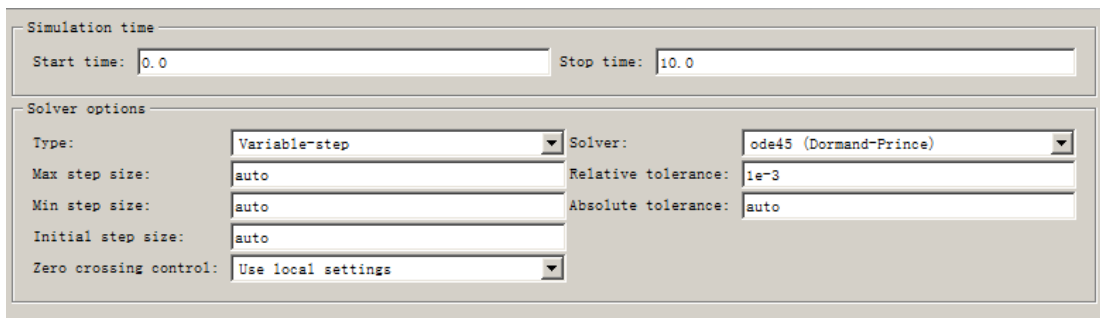


图 11.37 Solver 仿真参数的设置

2) Data Import/Export 参数

数据输入/输出选项页主要用于设置仿真数据的输入/输出, 如图 11.38 所示。Data Import/Export 页面包括 Load from workspace (从工作空间输入数据)、Save to workspace (将输出保存到工作空间) 和 Save options (保存选项)。其中,

- Load from workspace: 选择 “Input (输入)” 复选框可导入 MATLAB 工作空间内的数据, 其中时间变量一般定义为 t, 输入变量一般定义为 u。Initial state (初始状态) 用于导入状态初始值。
- Save to workspace: 该区域用于设置向 MATLAB 工作空间保存的数据名, 包括 Time (时间变量)、States (状态向量)、Output (输出变量) 和 Final states (系统稳态值)。
- Save options: 保存选项中可以设置的参数, 包括 Limit data points to last (限制仿真结果保存到 MATLAB 工作空间内的规模); Decimation (保存数据的间隔), 默认为 1, 即每个数据都保存下来; Format (返回数据的格式), 包括 Array (数组)、Struct (结构体)、Struct with time (带时间的结构体)。

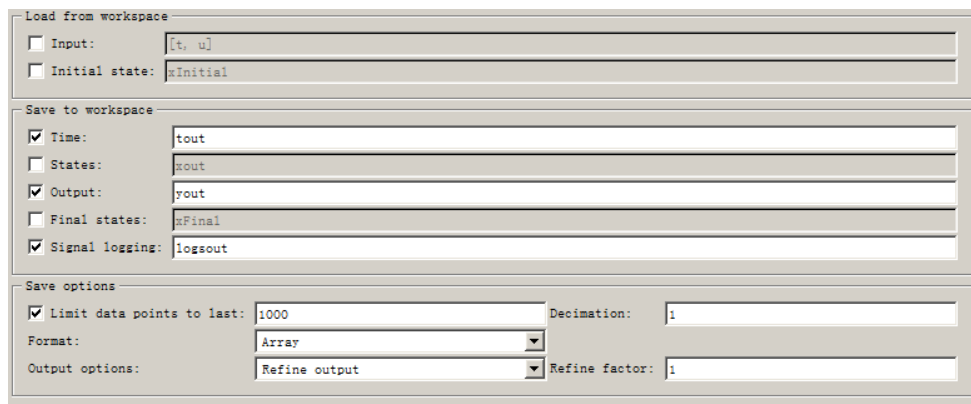


图 11.38 Solver 仿真参数的设置

3) Optimization 参数

优化参数主要设置仿真和代码生成的相关参数,可用于优化模型,提高模型的性能,如图 11.39 所示。其中:

- Simulation and code generation: 设置仿真与代码生成相关参数,包括 Block reduction optimization (合成模块)、Implement logic signals as boolean data (vs. double) (执行逻辑信号为布尔数据或者双精度数据)、Signal storage reuse (信号存储的重新分配)等选项。
- Code generation: 仅对代码生成设置相关参数。

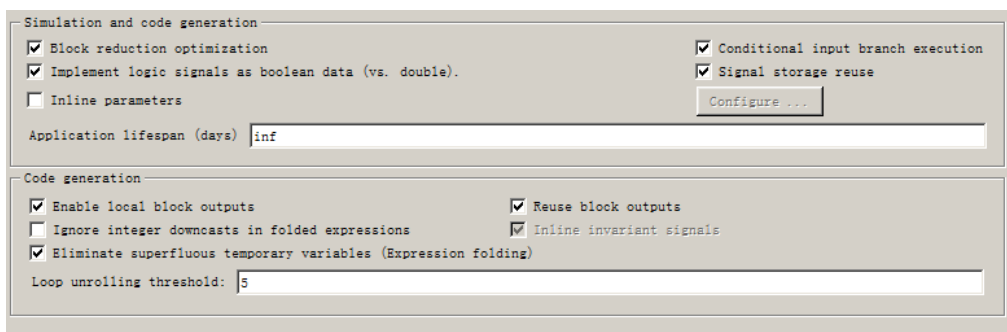


图 11.39 Optimization 仿真参数的设置

4) Diagnostics 参数

Diagnostics 仿真参数主要为系统模型的诊断参数,可以设置仿真过程中出现不正常状况时系统的处理方式。Diagnostics 参数设置包括 Solver、Sample Time、Data Integrity、Conversion、Connectivity、Compatibility 和 Model Reference 选项页。其中, Solver 选项页用于设置算法求解过程中出现异常的处理方法,如图 11.40 所示; Sample Time 选项页用于设置采样时间出现异常的处理方法; Data Integrity 选项页用于设置数据完整性异常的处理方法; Conversion 选项页用于设置数据转换相关的异常处理方法; Connectivity 选项页用于设置模块连接相关的异常处理方法; Compatibility 选项页用于设置版本兼容性相关的异常处理方法; Model Reference 选项页用于设置模块引用相关的异常处理方法。Diagnostics 各参数可以设置的仿真异常处理方法有 None (不做任何处理)、Warning (发出警告信息,程序正常运行)和 Error (错误,终止程序)。

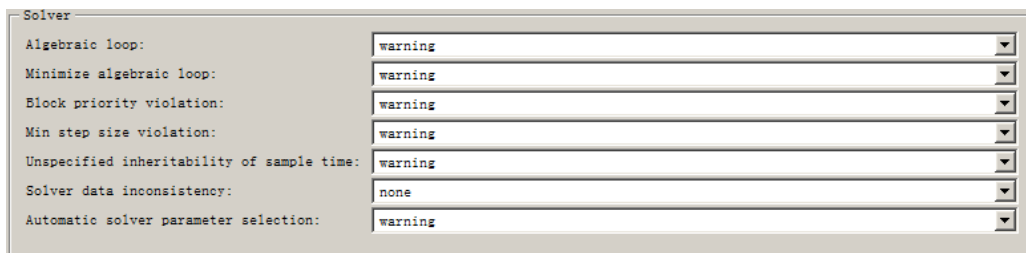


图 11.40 Diagnostics 仿真参数的设置

5) Hardware Implementation 参数

Hardware Implementation 仿真参数主要用于硬件相关参数设置，如图 11.41 所示。其中，

- Embedded hardware: 嵌入式硬件参数设置。
- Emulation hardware: 仿真式硬件参数设置。

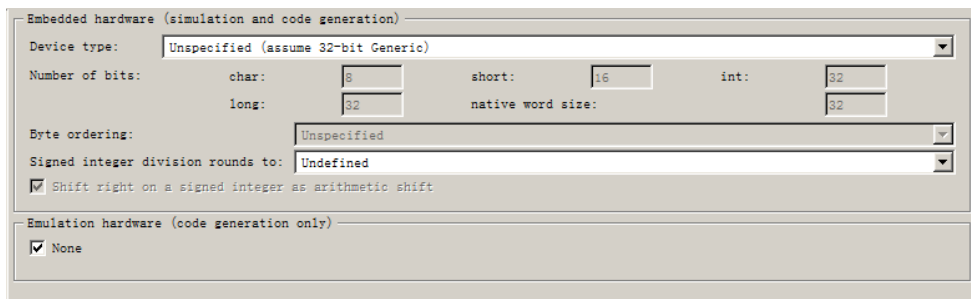


图 11.41 Hardware Implementation 仿真参数的设置

6) Model Referencing 参数

Model Referencing 仿真参数用于模型引用参数设置，如图 11.42 所示。其中，

- Rebuild options for all referenced models: 所有模型重构参数设置。
- Options for referencing this model: 引用模型参数设置。

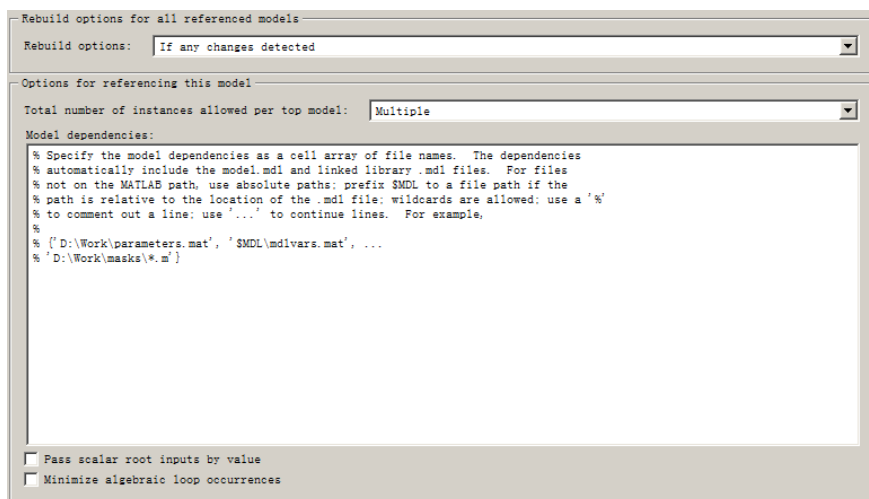


图 11.42 Model Referencing 仿真参数的设置

7) Real-Time Workshop 参数

Real-Time Workshop 仿真参数用于实时工作参数设置，如图 11.43 所示。其中，

- Target selection: 目标选择设置。
- Documentation: 控制文档生成设置。
- Build process: 构建构成设置。

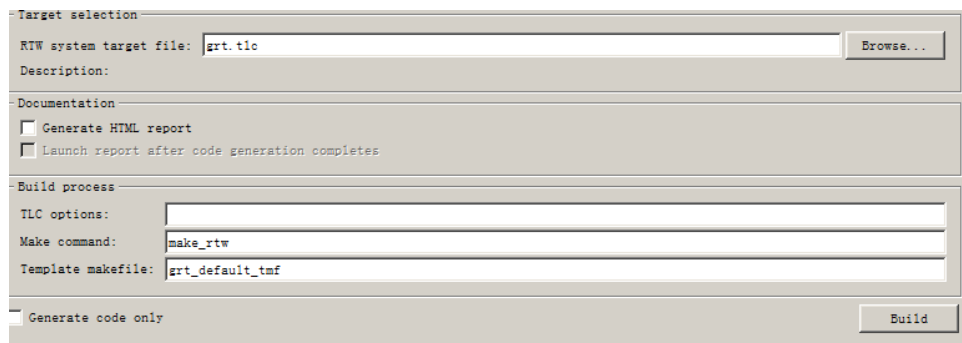



图 11.43 Real-Time Workshop 仿真参数的设置

11.3.5 仿真运行

在设置完各参数后，就可以对仿真模型进行运行了。系统模型的运行主要有以下几种方法。


(1) 单击模型编辑窗口工具栏中的运行工具 .

(2) 选择“Simulation”→“Start”命令。

(3) 使用快捷方式“Ctrl+T”。

(4) 利用 sim 命令进行动态系统仿真。函数 sim 的调用格式如下。

- sim(MPCobj,T,r): 运行系统模型框图名称为 MPCobj 的模型，参数 T 为系统仿真时间步长，r 为参考信号，与输出信号具有相同的列数。
- sim(MPCobj,T,r,v): 参数 v 指定观测信号的干扰信号，与观测信号具有相同的列数。
- sim(MPCobj,T,r,SimOptions): 参数 SimOptions 用于设置模型仿真参数，之前保存的模型如果已设置参数，将被新参数覆盖。

当模型运行完毕，需要对模型执行保存操作。可以通过“save”菜单或者工具栏工具  完成。


11.3.6 模型仿真举例


在本节中将向入门读者演示一个简单的仿真程序的生成。


【例 11.2】模型仿真的构建。


本例中将构建一个对正弦信号进行积分操作的简单例子，并显示积分后的波形。

(1) 新建模型，在模型库浏览窗口内选择需要的模块，具体如下。

 Sine Wave : 生成模型输入的正弦信号。

 Integrator : 对输入信号执行积分运算。

 Scope : 形象地显示输出信号。

 Mux : 输入信号和处理信号的混合。

(2) 将选择的模块添加到模型编辑窗口，并连接好信号线，如图 11-44 所示。

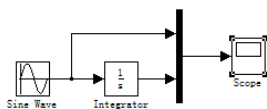


图 11.44 正弦积分仿真演示

(3) 本实例中各参数使用默认的参数。运行仿真程序，将在示波器上看到如图 11.45 所示的仿真结果。

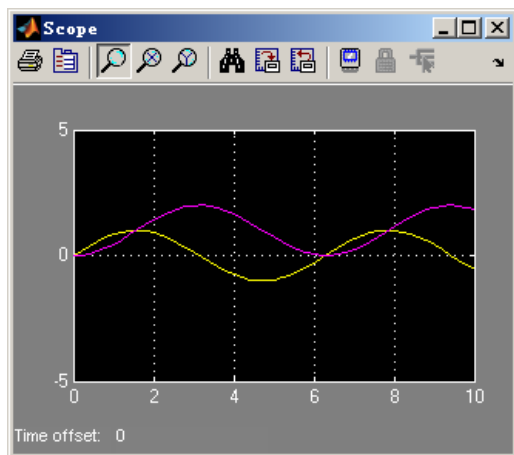


图 11.45 正弦积分仿真结果

(4) 模型保存。

11.4 本章小结

本章主要介绍了 Simulink 仿真的基础知识，通过本章的学习读者将对 Simulink 的使用有初步的了解。其中，包括对 Simulink 特点、Simulink 常用基本模块等的了解。本章 11.3 这一节重点讲述了如何利用 Simulink 进行建模仿真，通过学习读者将会对建模仿真的基本流程有深刻的体会。



第12章

统计工具箱

虽然 MATLAB 并不是专业的统计软件，但是其统计工具箱仍具有强大的功能，可以满足一般用户的需要。同时，在本书多章的介绍中，读者也可以充分发现 MATLAB 软件在数值分析、图像处理等方面同样具有强大的功能，而 MATLAB 各工具箱与 MATLAB 软件又是统一的整体，因而在使用 MATLAB 统计工具箱的同时我们也可以利用 MATLAB 的绘图、数据分析等功能，这在一定程度上使 MATLAB 统计工具箱更受用户的欢迎。本章将主要探讨 MATLAB 统计工具箱在假设测验、方差分析、线性回归、非线性回归、多元统计等较为常用的统计问题中的应用。

12.1 假设测验

假设测验是统计推断中的重要问题，可用于在总体未知的情况下，通过已知样本的特征对总体数据特征进行假设测验，用于推断在一定概率意义上是接受或者拒绝假设。假设测验的原理和步骤在此不再详细展开叙述，读者可以自行参考相应的统计书。本节主要围绕单个正态总体、两个正态总体的假设测验在 MATLAB 中的实现方式展开叙述。

12.1.1 单个正态总体的假设测验

本小节主要介绍常用的单个正态总体的假设测验，包括以下两种情况：总体标准差已知的均值 z 检验和总体标准差未知的均值 t 检验。

1. 总体标准差已知的均值 z 检验

在总体标准差已知的情况下，我们需要推断样本的均值是否等于总体的均值，其原假设为 $H_0: \mu = \mu_0$ ，备择假设为 $H_0: \mu \neq \mu_0$ 。在 MATLAB 中用于单个正态总体已知总体标准差的均值 z 检验的函数为 `ztest()`，其调用格式如下。

- $h = \text{ztest}(x, m, \text{sigma})$: x 为正态总体的样本数据， m 为需要检验的均值， sigma 为已知的总体标准差，使用默认的显著水平 0.05，返回假设测验的结果 h ，当 h 为 0 时表示在 0.05 的显著水平下，接受原假设。
- $h = \text{ztest}(x, m, \text{sigma}, \text{alpha})$: 参数 alpha 为假设测验的显著性水平。
- $[h, \text{sig}, \text{ci}, \text{zval}] = \text{ztest}(x, m, \text{sigma}, \text{alpha}, \text{tail})$: 参数 tail 用于设置假设测验的方式， tail 值='both' 为双边测验， tail 值='right' 为右侧测验， tail 值='left' 为左侧测验，测验后返回的参数除了反映拒绝或接受原假设的变量 h 外，还包括 sig (拒绝原假设的最小显著概率值)、 ci (真实均值的 $1-\text{alpha}$ 置信区间) 和 zval (Z 统计量的值)。

【例 12.1】 现有一批钢管，其长度符合标准差为 5cm 的正态分布，其平均值为 100cm，若随机抽取 10 根钢管，其长度分别为 101cm、98cm、105cm、103cm、97cm、99cm、100cm、101cm、108cm、98cm、试问在 0.05 的显著水平上，其是否符合均值为 100 cm?

原假设 $H_0: u=100$; 备择假设 $H_1: u \neq 100$ 。

```
x=[101 98 105 103 97 99 100 101 108 98];
h = ztest(x,100,0.05)
h =
    1
```

结果 $h=1$ 即拒绝原假设, 即认为钢管的均值可能不是 100cm。

2. 总体标准差未知的均值 t 检验

对于总体标准差未知的情况下, 对均值的假设检验可以通过函数 `ttest()` 来实现, 其调用格式如下。

- $h = \text{ttest}(x)$: 对正态总体 x 做均值为 0 的假设检验, 默认的显著性水平为 0.05, 返回假设检验的结果, $h=0$ 接受原假设, $h=1$ 拒绝原假设。
- $h = \text{ttest}(x,m)$: 对正态总体 x 做均值为 m 的假设检验, 默认的显著性水平为 0.05, 返回假设检验的结果, $h=0$ 接受原假设, $h=1$ 拒绝原假设。
- $h = \text{ttest}(x,m,\alpha)$: 对正态总体 x 做均值为 m 的假设检验, 默认的显著性水平为 α , 返回假设检验的结果, $h=0$ 接受原假设, $h=1$ 拒绝原假设。
- $h = \text{ttest}(x,m,\alpha,\text{tail})$: 参数 `tail` 用于设置假设检验的方式, `tail` 值='both' 为双边检验, `tail` 值='right' 为右侧检验, `tail` 值='left' 为左侧检验。
- $[h,p,ci] = \text{ttest}(\dots)$: t 检验返回参数 h (反映拒绝或接受原假设), p (拒绝原假设的最小显著概率值), ci (真实均值的 $1-\alpha$ 置信区间)。
- $[h,p,ci,stats] = \text{ttest}(\dots)$: t 检验返回参数 h (反映拒绝或接受原假设), p (拒绝原假设的最小显著概率值), ci (真实均值的 $1-\alpha$ 置信区间), $stats$ (t 测验的统计量, 其中包括 t 值、自由度和估计标准差)。

【例 12.2】对于例 12.1 中的问题, 如果总体标准差未知, 在 0.05 的显著水平上, 其是否符合均值为 100cm?

原假设 $H_0: u=100$; 备择假设 $H_1: u \neq 100$ 。

```
>> x=[101 98 105 103 97 99 100 101 108 98];
h = ttest(x,100,0.05)
h =
    0
p =
    0.385111165136970
ci =
    1.0e+002 *
    0.98521930986627    1.03478069013373
stats =
    tstat: 0.91287092917528
    df: 9
    sd: 3.46410161513775
```

结果 $h=0$ 即接受原假设, 即认为在 0.05 的水平上接受原假设。

12.1.2 两个正态总体的假设测验

这里主要介绍常用的总体标准差未知的均值 t 检验在 MATLAB 中的实现。

在两个正态总体的方差未知的情况下, 但需要假设量样本方差相等, 函数 `ttest2()` 可用于其均值的 t 检验, 函数 `ttest2()` 的调用格式如下。

- $[h,\text{significance},ci] = \text{ttest2}(x,y)$: x 和 y 为两正态总体的样本数据是否具有相同的均值, 其中 x 和 y 需假设来自具有相同标准差的总体, 但 x 和 y 样本的个数不要求一样, 函数返回的参数包括 h , $h=0$ 接受原假设, $h=1$ 拒绝原假设, `significance` 为拒绝原假设的最小显著

概率值, ci 为真实均值差异的 $1-\alpha$ 置信区间。

- `[h,significance,ci] = ttest2(x,y,alpha)`: 参数 `alpha` 用于设置 `t` 测验的显著水平。
- `[h,significance,ci,stats] = ttest2(x,y,alpha)`: 参数 `stats` 用于返回 `t` 测验的统计参数, 包括 `t` 测验的统计量, 其中包括 `t` 值、自由度和估计标准差。
- `[...] = ttest2(x,y,alpha,tail)`: 参数 `tail` 用于设置假设测验的方式, `tail` 值='both'为双边测验, `tail` 值='right'为右侧测验, `tail` 值='left'为左侧测验。

【例 12.3】 现有来自两正态总体的样本数据, 甲样本: 44.5, 45, 45.3, 46.1, 44.7, 48.6, 46.3, 47.5, 47.2; 乙样本: 44.2, 44.8, 46, 46.3, 48, 47.5, 45.4, 44.9, 47.4, 44.8, 45.6, 46.3, 47.5; 当甲、乙两样本总体的标准差相等的情况下, 甲和乙样本是否具有相同的均值?

原假设 $H_0: \mu_1 = \mu_2$; 备择假设 $H_1: \mu_1 \neq \mu_2$ 。

```
>> x=[44.5 45 45.3 46.1 44.7 48.6 46.3 47.5 47.2];
y=[44.2 44.8 46 46.3 48 47.5 45.4 44.9 47.4 44.8 45.6 46.3 47.5];
[h,significance,ci,stats] = ttest2(x,y,0.05)
h =
    0
significance =
    0.89003668026966
ci =
   -1.10459482206680    1.26356918104118
stats =
    tstat: 0.14003029411724
         df: 20
         sd: 1.30905133001551
```

执行上述程序, 可以看到结果为接受原假设, 即可认为在 95% 的显著水平上甲、乙两总体具有相同的均值。

12.2 方差分析

方差分析是用于科学研究的一种重要的统计分析方法。在实验研究中, 我们往往会设计不同的实验因素, 为了判断不同的因素是否引起实验结果差异我们需要做方差分析。因为实验结果的差异可能是不同实验因素造成的, 或者是外部的环境, 随机误差造成的。因而方差分析是检验实验处理间是否存在真实差异的有效手段。本节主要介绍利用 MATLAB 统计工具箱如何实现方差分析的操作, 包括单因素、双因素和多因素的方差分析。下面具体介绍这些内容。

12.2.1 单因素方差分析

当实验仅有一个因素, 即单因素实验, 我们仅需要研究单因素的不同水平是否对实验结果的差异有显著的影响, 此时我们需要做单因素方差分析。在 MATLAB 中实现单因素方差分析的函数为 `anova1()`, 其调用格式如下。

- `p = anova1(X)`: 数据矩阵 `X` 为 $m \times n$ 的矩阵, m 为因素每个水平的实验重复观测数, n 为实验因素的水平, 对矩阵 `X` 的列数据进行均值比较, 零假设为各列的均值相等, 所有数据来自同一个整体, 其因素的水平没有对结果造成影响, 如果返回的概率值 `p` 接近 0, 即不接受零假设, 各列的均值的差异是由实验因素的差异造成的。
- `p = anova1(X,group)`: 参数 `group` 可为字符数组或者元胞数组, 用于指定数据矩阵 `X` 的分组情况, 相应的矩阵 `X` 每列在参数 `group` 中都对应一个标签, 具有相同 `group` 值的矩阵 `X` 中的列元素即为同一组。

- `p = anova1(X,group,'displayopt')`: 参数 `displayopt` 用于控制方差分析表和盒形图的显示, 默认状态下参数 `displayopt` 值为 `on`, 显示方差分析表和盒形图, 参数值为 `off` 即表示不显示。
- `[p,table] = anova1(...)`: 返回方差分析的概率值 `p` 和方差分析表 `table`。
- `[p,table,stats] = anova1(...)`: 返回结构体变量 `stats` 用于进一步的多重比较, 一般在方差分析后如果各因素处理间的均值不完全相同, 还需要确定哪些处理间存在差异, 这时需要利用多重比较。
- 在 MATLAB 统计工具箱中还提供了函数 `multcompare()` 用于多重比较, 其调用格式如下。
- `c = multcompare(stats)`: 参数 `stats` 为通过方差分析函数返回的结构统计参数, 返回两两比较的结果参数 `c`, `c` 为多行 5 列的数据矩阵, 其各列依次为多重比较的组号, 比较的两个组的均值差和均值差的置信区间。多重比较同时还返回一个交互式的图形, 该图形以符号标志了每一组的均值, 并通过线段标出了均值的置信区间。
- `c = multcompare(stats, param1, val1, param2, val2,...)`: 设置参数 `param1` 的值为 `val1`, 参数 `param2` 的值为 `val2`。可以设置的多重比较的参数包括 `alpha` (用于指定置信水平)、`displayopt` (是否显示图形)、`ctype` (多重比较临界值的类型)、`dimension` (用于指定比较的因素) 和 `estimate` (用于指定要比较的估计)。
- `[c,m] = multcompare(...)`: 参数 `m` 为两列数据矩阵, 其第一列为每组均值的估计值, 第二列为每组的标准误差。
- `[c,m,h] = multcompare(...)`: 返回的参数 `h` 为交互式图的句柄值。
- `[c,m,h,gnames] = multcompare(...)`: 返回参数 `gnames` 为元胞数组, 为对应各组组名的行数据。

【例 12.4】 有 A、B、C、D、E、F6 个小麦品种产量的比较试验, 设置标准品种 CK, 采用 3 次重复的对比设计, 所得产量结果如表 12.1 所示。

表 12.1 6 个小麦品种产量结果分析

品种	各重复的产量(kg)		
	I	II	III
A	533	565	525
B	580	600	575
C	525	500	510
D	600	615	590
E	570	575	565
F	650	661	643
CK	500	510	513

编写程序, 进行方差分析。

```
>> X=[533 580 525 600 570 650 500;565 600 500 615 575 661 510; 525 575 510 590 565 643
513];
group={'A','B','C','D','E','F','CK'};
[p,table,stats]=anova1(X,group)
p =
    1.032806173917322e-008
table =
    Columns 1 through 3
    'Source'    'SS'                                'df'
    'Columns'   [4.771514285714286e+004]          [ 6]
    'Error'     [2.186666666666667e+003]        [14]
    'Total'     [4.990180952380953e+004]        [20]
    Columns 4 through 5
    'MS'                                'F'
```



```

[7.952523809523809e+003]    [50.91554878048769]
[1.561904761904766e+002]    []
                                []
Column 6
'Prob>F'
[1.032806173917322e-008]
                                []
                                []
stats =
    gnames: {7x1 cell}
           n: [3 3 3 3 3 3]
    source: 'anova1'
     means: [1x7 double]
        df: 14
         s: 12.49761882081849

```

通过方差分析发现处理间存在差异, 进而进行多重比较。

```

>> [c,m,h,gnames] = multcompare(stats)
c =
1.0e+002 *
Columns 1 through 3
    0.010000000000000    0.020000000000000    -0.78843338037279
    0.010000000000000    0.030000000000000    -0.05510004703946
    0.010000000000000    0.040000000000000    -0.95510004703946
    0.010000000000000    0.050000000000000    -0.63843338037279
    0.010000000000000    0.060000000000000    -1.45176671370613
    0.010000000000000    0.070000000000000    -0.01510004703946
    0.020000000000000    0.030000000000000    0.38489995296054
    0.020000000000000    0.040000000000000    -0.51510004703946
    0.020000000000000    0.050000000000000    -0.19843338037279
    0.020000000000000    0.060000000000000    -1.01176671370613
    0.020000000000000    0.070000000000000    0.42489995296054
    0.030000000000000    0.040000000000000    -1.24843338037279
    0.030000000000000    0.050000000000000    -0.93176671370613
    0.030000000000000    0.060000000000000    -1.74510004703946
    0.030000000000000    0.070000000000000    -0.30843338037279
    0.040000000000000    0.050000000000000    -0.03176671370613
    0.040000000000000    0.060000000000000    -0.84510004703946
    0.040000000000000    0.070000000000000    0.59156661962721
    0.050000000000000    0.060000000000000    -1.16176671370613
    0.050000000000000    0.070000000000000    0.27489995296054
    0.060000000000000    0.070000000000000    1.08823328629387
Columns 4 through 5
    -0.440000000000000    -0.09156661962721
    0.293333333333333    0.64176671370613
    -0.606666666666667    -0.25823328629387
    -0.290000000000000    0.05843338037279
    -1.103333333333333    -0.75489995296054
    0.333333333333333    0.68176671370613
    0.733333333333333    1.08176671370613
    -0.166666666666667    0.18176671370613
    0.150000000000000    0.49843338037279
    -0.663333333333333    -0.31489995296054
    0.773333333333333    1.12176671370613
    -0.900000000000000    -0.55156661962721
    -0.583333333333333    -0.23489995296054
    -1.396666666666667    -1.04823328629387
    0.040000000000000    0.38843338037279
    0.316666666666667    0.66510004703946
    -0.496666666666667    -0.14823328629387
    0.940000000000000    1.28843338037279
    -0.813333333333333    -0.46489995296054

```

```

0.623333333333333 0.97176671370613
1.43666666666667 1.78510004703946
m =
1.0e+002 *
5.41000000000000 0.07215503590429
5.85000000000000 0.07215503590429
5.11666666666667 0.07215503590429
6.01666666666667 0.07215503590429
5.70000000000000 0.07215503590429
6.51333333333333 0.07215503590429
5.07666666666667 0.07215503590429
h =
2
gnames =
'A'
'B'
'C'
'D'
'E'
'F'
'CK'

```

执行上述程序将生成如图 12.1 所示的单因素方差分析表, 如图 12.2 所示为方差分析的盒形图, 如图 12.3 所示为交互式的多重比较图。

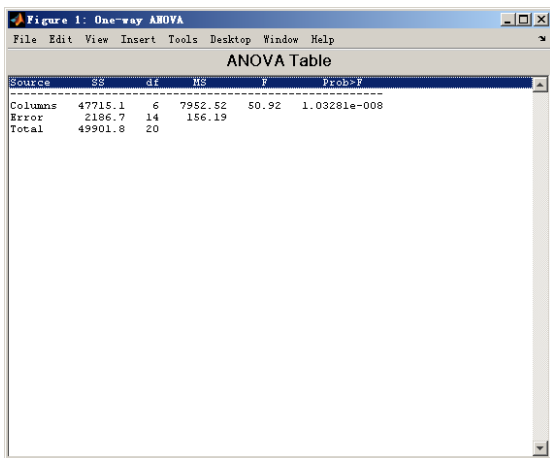


图 12.1 单因素方差分析表

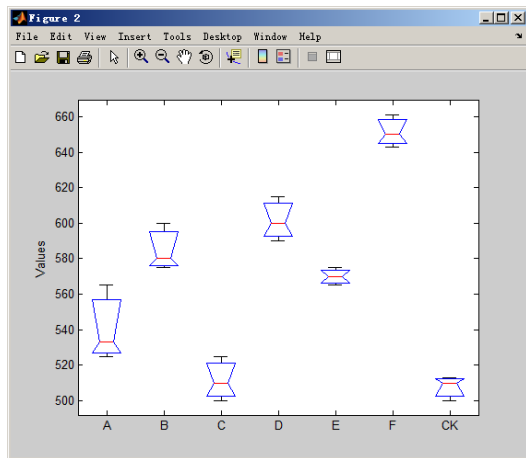


图 12.2 方差分析的盒形图

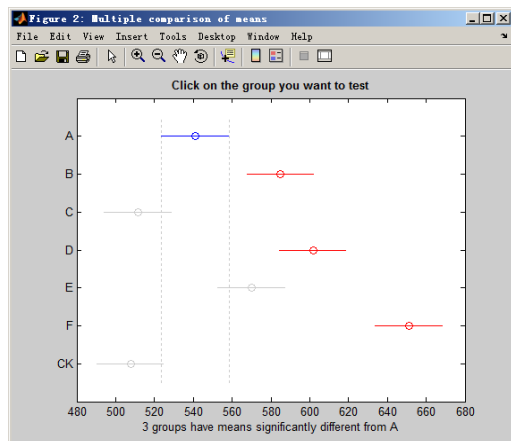


图 12.3 交互式的多重比较图

12.2.2 双因素方差分析

在实验设计中如果涉及两个实验因素，需要进行双因素的方差分析。MATLAB 统计工具箱中还提供了函数 `anova2()` 用于双因素的方差分析，其调用格式如下。

- `p = anova2(X, reps)`: 参数 `X` 为进行方差分析的样本观测，如果实验设计中有 `A` 和 `B` 两个因素，其中数据 `X` 的每一列对应 `A` 因素的一个水平，每一行对应 `B` 因素的一个水平；如果实验每一因素水平下进行了重复实验，参数 `reps` 为实验的重复数，默认情况下为 1；当 `reps` 值为 1 的时候返回的 `p` 值为两个元素的向量，分别为检验 `H0A`: 因素 `A` 的所有样本来自相同的总体，`H0B`: 因素 `B` 的所有样本来自相同的总体，当 `reps` 值大于 1 的时候，返回的 `p` 值还包括检验 `HAB`，`A` 和 `B` 实验因素的交互影响。
- `p = anova2(X, reps, 'displayopt')`: 参数 `displayopt` 用于控制方差分析表的显示。
- `[p, table] = anova2(...)`: 参数 `table` 返回方差分析表。
- `[p, table, stats] = anova2(...)`: 参数 `stats` 返回结构体变量，用于进一步的多重比较。

【例 12.5】 水稻品种（因素 `A`）和密度（因素 `B`）实验的产量结果方差分析，实验数据如表 12.2 所示。

表 12.2 水稻品种（因素 `A`）和密度（因素 `B`）实验的产量结果

处理	重复		
	I	II	III
A_1B_1	40	38	42
A_2B_1	46	42	44
A_3B_1	47	43	45
A_1B_2	42	44	45
A_2B_2	48	47	46
A_3B_2	50	48	49

以下程序根据表 12.2 的水稻品种（因素 `A`）和密度（因素 `B`）实验的产量结果进行方差分析。

```
X=[40 46 47;38 42 43;42 44 45;42 48 50; 44 47 48;45 46 49];
[p,table,stats] =anova2(X,3)           %方差分析
p =
    0.00046305657867    0.00064188820333    0.86833040618676
table =
Columns 1 through 3
    'Source'    'SS'
    'Columns'   [    84.7777777777794]   [    2]
    'Rows'      [    56.8888888888872]   [    1]
    'Interaction' [    0.77777777778587]   [    2]
    'Error'      [    32.66666666667152]  [   12]
    'Total'      [1.751111111111241e+002] [   17]
Columns 4 through 5
    'MS'    'F'
    [42.38888888888897]   [15.57142857142629]
    [56.88888888888872]   [20.89795918367030]
    [ 0.38888888889294]   [ 0.14285714285861]
    [ 2.72222222222263]   [ ]
    [ ]
Column 6
    'Prob>F'
    [4.630565786718499e-004]
    [6.418882033253359e-004]
    [ 0.86833040618676]
    [ ]
    [ ]
stats =
```

```

source: 'anova2'
sigmasq: 2.72222222222263
colmeans: [41.8333333333333 45.5000000000000 47]
coln: 6
rowmeans: [43 46.5555555555555]
rown: 9
inter: 1
pval: 0.86833040618676
df: 12

```

执行上述程序，将生成如图 12.4 所示的双因素方差分析表。以上结果显示，A 和 B 因素对实验结果有显著的影响，但是 A 和 B 交互没有显著的影响。

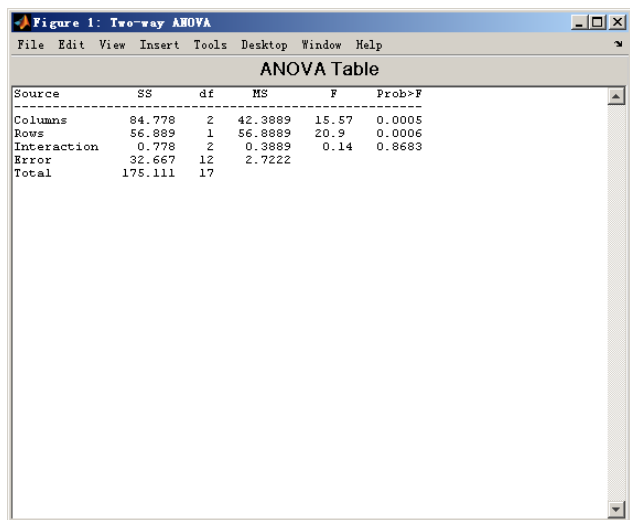


图 12.4 双因素方差分析表

12.2.3 多因素方差分析

对涉及两个以上因素的实验需要进行多因素方差分析，在 MATLAB 中进行多因素方差分析的函数为 `anovan()`，其调用格式如下。

- `p = anovan(x,group)`: 参数 `x` 为进行方差分析的多因素的实验观测数据，`group` 用于指定各因素的分组情况，返回值 `p` 用于检验各因素的水平是否具有显著差异，向量 `p` 的大小为实验的因素数。
- `p = anovan(x,group,'Param1',val1,'Param2',val2,...)`: 设置参数 `param1` 的值为 `val1`，参数 `param2` 的值为 `val2`，可以设置的多重比较的参数具体参考 MATLAB 的帮助文档。
- `[p,table] = anovan(...)`: 参数 `table` 返回方差分析表。
- `[p,table,stats] = anovan(...)`: 返回的结构体 `stats` 用于多重比较。

【例 12.6】多因素方差分析。

```

>> y = [52 57 45 44 53 57 45 44]';
g1 = [1 2 1 2 1 2 1 2];
g2 = {'hi','hi','lo','lo','hi','hi','lo','lo'};
g3 = {'may','may','may','may','june','june','june','june'};
[p,table,stats]= anovan(y, {g1 g2 g3})           %多因素的方差分析
p =
    0.27706916734759
    0.00181195233313
    0.86619369222637
table =
Columns 1 through 3

```

```

'Source'      'Sum Sq.'      'd.f.'
'X1'          [ 6.12500000000011] [ 1]
'X2'          [2.101250000000001e+002] [ 1]
'X3'          [ 0.12500000000009] [ 1]
'Error'       [ 15.4999999999994] [ 4]
'Total'       [2.31875000000000e+002] [ 7]
Columns 4 through 5
'Singular?'    'Mean Sq.'
[ 0]          [ 6.12500000000011]
[ 0]          [2.101250000000001e+002]
[ 0]          [ 0.12500000000009]
[ 0]          [ 3.87499999999999]
[ 0]          []
Columns 6 through 7
'F'            'Prob>F'
[ 1.58064516129036] [0.27706916734759]
[54.22580645161312] [0.00181195233313]
[ 0.03225806451615] [0.86619369222637]
[]                []
[]                []
stats =
    source: 'anovan'
    resid: [8x1 double]
    coeffs: [7x1 double]
    Rtr: [4x4 double]
    rowbasis: [4x7 double]
    dfe: 4
    mse: 3.87499999999999
    nullproject: [7x4 double]
    terms: [3x3 double]
    nlevels: [3x1 double]
    termcols: [4x1 double]
    coeffnames: {7x1 cell}
    vars: [7x3 double]
    varnames: {3x1 cell}
    grpnames: {3x1 cell}
    ems: []
    denom: []
    dfdenom: []
    msdenom: []
    varest: []
    varci: []
    txtdenom: []
    txtems: []
    rtnames: []

```

执行上述程序，将生成如图 12.5 所示的多因素方差分析表。

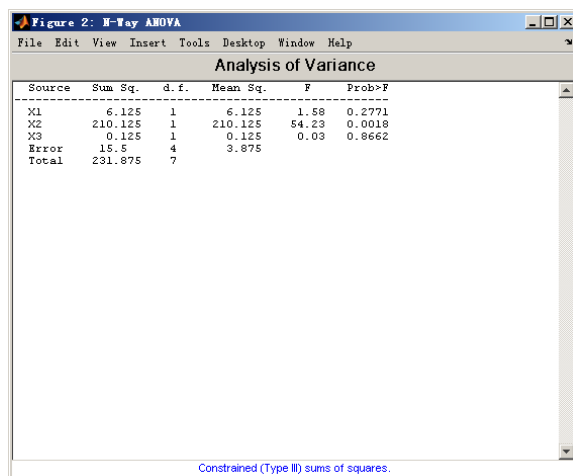


图 12.5 多因素方差分析表

12.3 线性回归

在许多实际的问题中，常常需要研究两个变量之间的关系，并构建简单的模型拟合观测数据，其中线性回归是最常用的分析方法，且一些指数、幂函数等关系也可以化解为简单的线性回归关系。在线性回归分析中通过需要考虑自变量和因变量，同时按照自变量的个数，线性回归分析可以分为一元线性回归和多元线性回归。

在 MATLAB 中用于线性回归的函数为 `regress()`，该函数可以同时进行一元和多元线性回归分析，其调用格式如下。

- `b = regress(y,X)`: 参数 `y` 为 $n \times 1$ 的矩阵，为线性回归的因变量，参数 `X` 为 $n \times p$ 的矩阵，为线性回归的自变量，一般需要在原始自变量左边加一列全为 1 的数据，代表线性模型的常数项，函数返回线性回归系数的估计值 `b`，其中第一个值为线性模型的截距项，第二个值及其以后的值为模型的回归系数。
- `[b,bint,r,rint,stats] = regress(y,X)`: 返回参数 `bint` 为线性模型回归系数估计值的置信度为 95% 的置信区间，参数 `r` 为拟合的线性模型的残差（线性回归模型的真实值减去预测值），参数 `rint` 为模型残差的 95% 置信区间，参数 `stats` 为包含 4 个元素的向量，用于反应拟合的线性模型的统计特征，包括模型决定系数 `R2`，`F` 统计量，显著性概率 `P` 值和模型误差的方差。
- `[b,bint,r,rint,stats] = regress(y,X,alpha)`: 参数 `alpha` 用于设置模型的置信水平。

【例 12.7】 利用函数 `regress()` 进行一元线性回归分析。

如表 12.3 所示为产量与积温的一元线性回归分析。

表 12.3 产量与积温的一元线性回归分析

x(积温)	1616	1532	1732	1423	1574	1602	1654	1448	1567	1668
y(产量)	435	365	502	287	389	423	460	301	384	468

以下程序根据表 12.3 中的数据，建立产量与积温的一元线性回归分析。

```
>> x=[1616 1532 1732 1423 1574 1602 1654 1448 1567 1668];
y=[435 365 502 287 389 423 460 301 384 468];
X=[ones(size(x,2),1),x'];
[b,bint,r,rint,stats] = regress(y',X)           %一元线性回归分析
b =
    1.0e+002 *
   -7.54938021108051
    0.00731119133225
bint =
    1.0e+002 *
   -8.38924176686158   -6.70951865529944
    0.00678104633401    0.00784133633048
r =
    8.44950181707321
   -0.13649099205912
   -9.36031763698213
    1.55549452942392
   -6.84349458749296
    6.68516968221786
    5.66697475453788
   -2.72248380119146
   -6.72566065492060
```

```

3.43130688939323
rint =
-5.00119550925157 21.90019914339800
-15.33654216086286 15.06356017674461
-19.34743264544720 0.62679737148294
-10.94747312697447 14.05846218582232
-21.09812926381239 7.41114008882647
-7.58614976661089 20.95648913104661
-8.39448855435050 19.72843806342625
-15.98881914446540 10.54385154208248
-21.00536975541722 7.55404844557602
-10.92264318840674 17.78525696719320
stats =
1.0e+003 *
Columns 1 through 3
0.00099215196272 1.01136315024839 0.00000000000104
Column 4
0.04368845390296

```

执行上述程序，将生成如图 12.6 所示的一元线性回归的散点图和回归模型的图。

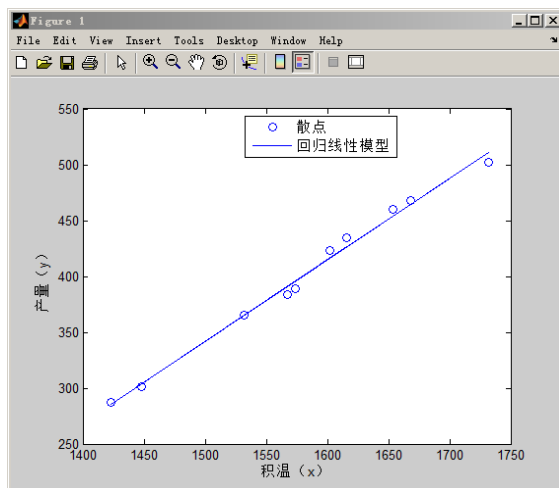


图 12.6 一元线性回归的散点图和回归模型

【例 12.8】利用函数 regress()进行多元线性回归分析。

如图 12.4 所示为需求量与价格和收入的多元线性回归分析。

表 12.4 需求量与价格和收入的多元线性回归分析

需求量	价格	收入
59190.0	23.56	76200.0
65450.0	24.44	91200.0
62360.0	32.07	106700.0
64700.0	32.46	111600.0
67400.0	31.15	119000.0
64440.0	34.14	129200.0
68000.0	35.3	143400.0
72400.0	38.7	159600.0
75710.0	39.63	180000.0
70680.0	46.68	193000.0

以下程序根据表 12.4 所示的数据构建需求量与价格和收入的多元线性回归分析。

```

>> y=[59190 65450 62360 64700 67400 64440 68000 72400 75710 70680];
x=[23.56 24.44 32.07 32.46 31.15 34.14 35.3 38.7 39.63 46.68;

```

```

76200 91200    106700 111600 119000 129200 143400 159600 180000 193000];
X=[ones(size(x,2),1),x'];
[b,bint,r,rint,stats] = regress(y',X)           %利用函数 regress() 进行多元线性回归分析
b =
    1.0e+004 *
    6.26509284667073
   -0.09790570096564
    0.00002861815879
bint =
    1.0e+004 *
    5.31616677732703    7.21401891601443
   -0.17352266565249   -0.02228873627879
    0.00001481342919    0.00004242288840
r =
    1.0e+003 *
   -2.20138231821998
    0.62746403157335
    0.57185440259071
    1.89139685557997
    1.19108842236926
   -1.76058331547698
   -1.12865573264896
    1.96399637598221
    0.34641500152482
   -1.50159372327451
rint =
    1.0e+003 *
   -5.10202397628720    0.69925933984723
   -2.72595618011617    3.98088424326287
   -2.92913810927725    4.07284691445867
   -1.42627375334888    5.20906746450881
   -2.81022350773960    5.19240035247812
   -5.57969515175542    2.05852852080146
   -5.12937325753759    2.87206179223968
   -1.62682789728223    5.55482064924665
   -2.59952197434898    3.29235197739862
   -4.19695228889270    1.19376484234369
stats =
    1.0e+006 *
Columns 1 through 3
    0.00000090221846    0.00003229407689    0.00000000029235
Column 4
    3.02406751090850

```

函数 regstats()用于线性回归诊断，其调用格式如下。

- regstats(responses,DATA): 参数 responses 为因变量的观测矩阵，是 $n \times 1$ 的列向量，参数 DATA 为自变量的观测矩阵，默认情况下会自动在数据的左边加上一列全为 1 的数据，不需像函数 regress 一样要用户自己添加，调用该函数后将生成选择回归统计参数的 GUI 界面，如图 12.7 所示，用户从其中选择需要计算的统计参数后，将生成相应的统计参数，并导入 MATLAB 工作空间中。
- regstats(responses,DATA,model): 参数 model 用于控制回归模型的类型，包括 linear (带常数项的回归模型，默认)、interaction (带常数项、线性项、交叉项的回归模型)、quadratic (带常数项、线性项、交叉项、平方项的回归模型) 和 purequadratic (带常数项、线性项、平方项的回归模型)。
- stats = regstats(...): 该函数调用格式将线性回归诊断参数保存到参数 stats 中，不生成统计参数选择的 GUI 界面。
- stats = regstats(responses,data,model,whichstats): 参数 whichstats 用于设置需要返回的

线性回归统计参数。

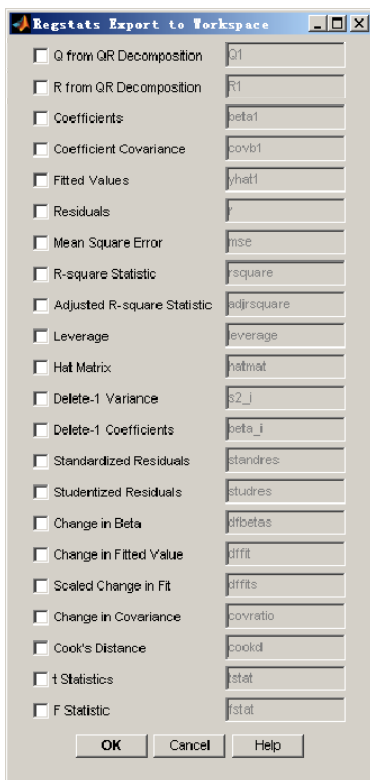


图 12.7 函数 regstats()用于线性回归诊断参数选择 GUI

【例 12.9】利用函数 regstats()进行线性回归分析。

```
>> load hald
stats=regstats(heat,ingredients)           %利用函数 regstats() 进行线性回归分析
stats =
    source: 'regstats'
         Q: [13x5 double]
         R: [5x5 double]
        beta: [5x1 double]
       covb: [5x5 double]
       yhat: [13x1 double]
         r: [13x1 double]
        mse: 5.98295491881238
    rsquare: 0.98237562040768
adjrsquare: 0.97136038316248
leverage: [13x1 double]
    hatmat: [13x13 double]
       s2_i: [13x1 double]
      beta_i: [5x13 double]
   standres: [13x1 double]
    studres: [13x1 double]
   dfbetas: [5x13 double]
     dffit: [13x1 double]
    dffits: [13x1 double]
   covratio: [13x1 double]
     cookd: [13x1 double]
      tstat: [1x1 struct]
      fstat: [1x1 struct]
```

在回归分析中如果存在异常数据点将会对最终的回归模型造成很大的影响，稳健回归采用加

权最小二乘法估计回归参数，所构建的模型受异常数据点的影响较小。在 MATLAB 统计工具箱中进行稳健回归的函数为 `robustfit()`，其调用格式如下。

- `b = robustfit(X,Y)`: 参数 X 为自变量的观测矩阵，用户不需要在自变量 X 的前面加上全为 1 的数据，自动添加常数项，参数 Y 为因变量的观测矩阵，返回稳健回归模型的估计系数 b 。
- `[b,stats] = robustfit(X,Y)`: 参数 $stats$ 为结构变量，用于存储各项统计参数。
- `[b,stats] = robustfit(X,Y,'wfun',tune,'const')`: 参数 $wfun$ 为稳健回归的加权函数，参数 $tune$ 为调节常数，参数 $const$ 用于控制模型是否包含常数项。

【例 12.10】 利用函数 `robustfit()` 进行稳健回归分析。

```
>> load hald
[b,stats]=robustfit(ingredients,heat)           %利用函数 robustfit() 进行稳健回归分析
b =
    60.03844626248139
    1.57584319838077
    0.53268587541473
    0.13416098741772
   -0.12000178056977
stats =
    ols_s: 2.44600795559057
   robust_s: 2.54374261068421
    mad_s: 3.56530279604628
         s: 2.54374261068421
   resid: [13x1 double]
   rstud: [13x1 double]
        se: [5x1 double]
coeffcorr: [5x5 double]
         t: [5x1 double]
         p: [5x1 double]
         w: [13x1 double]
         R: [5x5 double]
        dfe: 8
         h: [13x1 double]
```

同时对于一些简单的非线性函数，如指数函数、幂函数等可以转换为线性函数进行求解，各种函数的转换方式如表 12.5 所示。

表 12.5 可线性化的回归分析

原回归方程	转换过程				转换后的线性方程
	u	x'	c	d	
$y=ae^{bx}$	$u=\ln y$	$x'=x$	$c=\ln a$	$d=b$	$u=c+dx'$
$y=a+b\ln x$	$u=y$	$x'=\ln x$	$c=a$	$d=b$	$u=c+dx'$
$y=ax^b$	$u=\ln y$	$x'=\ln x$	$c=\ln a$	$d=b$	$u=c+dx'$

12.4 非线性回归

对于一些无法转换为线性回归问题的非线性问题，在 MATLAB 中利用函数 `nlinfit()` 来求解，其调用格式如下。

- `beta = nlinfit(X,y,fun,beta0)`: 输入参数 X 为自变量的观测矩阵， y 为因变量的观测矩阵，参数 fun 为非线性方程的函数名， $beta0$ 为用户设置的初始估计的回归模型的参数，返回非线性回归方程的估计参数 $beta$ 。
- `[beta,r,J] = nlinfit(X,y,fun,beta0)`: 参数 r 为非线性回归方程估计的残差，参数 J 为雅克比矩阵。

- `[...] = nlinfit(X, y, fun, beta0, options)`: 参数 `options` 用于非线性回归模型参数估计算法的相关参数设置。
- 函数 `nlpredci()` 可用于计算非线性回归模型的预测值, 其调用格式如下。
- `ypred = nlpredci(fun,inputs,beta,r,J)`: 根据非线性拟合函数 `nlinfit()` 获得的输出参数 `beta`, `r` 和 `J` 获取函数 `fun` 在 `inputs` 处的函数预测值 `ypred`。
- `[ypred,delta] = nlpredci(FUN,inputs,beta,r,J)`: 返回的参数 `delta` 为预测值 95% 置信区间的一半, 即预测值的置信区间为 $[ypred - delta, ypred + delta]$ 。
- `ypred = nlpredci(FUN,inputs,beta,r,J,alpha,'simopt','predopt')`: 输入参数 `alpha` 为置信区间, `simopt` 为可以设置的值为 `on` 或 `off` (默认), 分别表示同步或不同步的置信区间, `predopt` 值可为 `curve` (默认), 表示计算输入变量函数预测值的置信区间, 值为 `observation` 表示新响应值的置信区间。
- 函数 `nlparci()` 可用于计算非线性回归模型估计的回归系数的置信区间, 其调用格式如下。
- `ci = nlparci(beta,resid,J)`: 输入参数 `beta`、`resid` 和 `J` 为利用函数 `nlinfit()` 计算所得的非线性模型的结果, 返回的参数 `ci` 为模型估计的系数的置信区间。
- `ci = nlparci(beta,resid,J,alpha)`: 参数 `alpha` 为置信水平。

【例 12.11】 利用函数 `nlpredci()` 进行非线性回归分析。

```
clear
load data;
beta0=[1.1 2.8 4.6];
[beta,r,J] = nlinfit(x,y,@nlinfittest,beta0);
disp(beta)
    1.25688465913389    3.000000000000000    4.95368539023927
[ypred,delta] = nlpredci(@nlinfittest,x,beta,r,J); %利用函数 nlpredci() 进行非线性回归分析
ci = nlparci(beta,r,J)
ci =
    1.25665313492896    1.25711618333882
    3.00000000000000    3.000000000000001
    4.95350118818804    4.95386959229049
```

函数 `nlintool()` 可调用交互式的图形工具箱进行非线性回归, 其调用格式如下。

- `nlintool(x,y,fun,beta0)`: 数据 `x` 为自变量, `y` 为因变量, 参数 `fun` 为拟合的非线性回归模型, 参数 `beta0` 为用户估计的模型估计参数, 调用该格式后将生成非线性回归交互式的图形工具箱。
- `nlintool(x,y,fun,beta0,alpha)`: 参数 `alpha` 用于指定非线性回归的置信水平。
- `nlintool(x,y,fun,beta0,alpha,'xname','yname')`: 参数 `xname` 和 `yname` 分别设置坐标轴的字符串名称。

【例 12.12】 利用交互式工具进行非线性回归分析。

```
clear
load data;
beta0=[1.1 2.8 4.6];
nlintool(x,y,@nlinfittest,beta0) %利用交互式工具进行非线性回归分析
```

执行上述程序将弹出交互式工具进行非线性回归分析, 如图 12.8 所示。单击非线性回归交互式工具窗口中的“Export”按钮, 将弹出如图 12.9 所示的非线性回归模型统计参数设置对话框。选择需要计算的统计量将输出到 MATLAB 工作空间内。同时, 非线性回归分析的交互式工具窗口中的菜单包括如下几个。

- “File” 菜单: 主要为常用的文件操作, 包括文件的打开、保存, 数据的导入、导出等。
- “Edit” 菜单: 包括常用的文件操作, 复制、粘贴, 图形窗口的相关属性设置。
- “View” 菜单: 用于一些图形窗口工具的设置。

- “Bounds” 菜单：用于设置非线性回归算法的一些参数设置。
- “Insert” 菜单：用于在图形窗口中插入坐标轴和相关的图形。
- “Tools” 菜单：选择图形窗口的一些编辑工具。
- “Desktop” 菜单：控制交互式工具窗口的位置。
- “Window” 菜单：控制打开的窗口。
- “Help” 菜单：显示帮助信息。

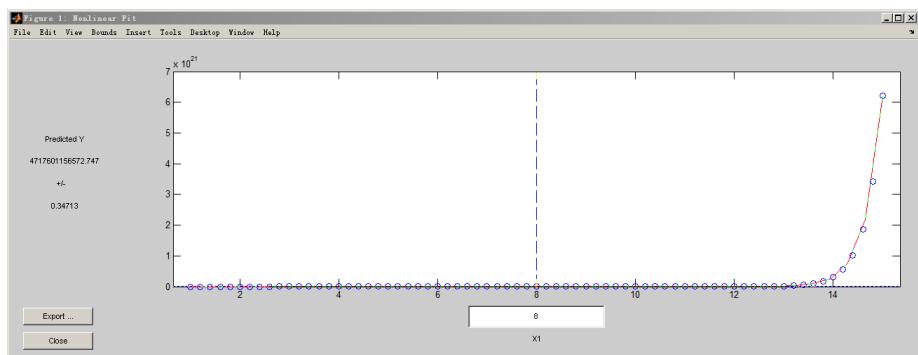


图 12.8 利用交互式工具进行非线性回归分析

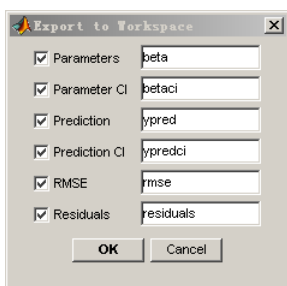


图 12.9 非线性回归统计参数分析

12.5 多元统计

多元统计也是较为常用的统计分析方法，本节主要介绍判别分析、聚类分析、主成分分析和因子分析。下面具体介绍如何利用 MATLAB 实现这些多元统计的操作。

12.5.1 判别分析

判别分析是根据已知分类的数据特征建立判别分类规则，然后利用建立的数据分类规则对未知分类的数据进行分类。在 MATLAB 中利用 `classify()` 函数可以进行判别分析，其调用格式如下。

- `class = classify(sample, training, group)`: 参数 `sample` 为待进行判别分析的数据，参数 `training` 为已知分类的样本数据，`sample` 与 `training` 应具有相同的列数，参数 `group` 为已知分类的样本数据的分类结果，`group` 与 `training` 具有相同的行数，返回分类后的结果 `class`，即按照已知样本的判别规则未知样本数据 `sample` 各行数据的类别，因而矩阵 `sample` 与 `class` 应具有相同的行数。
- `class = classify(sample, training, group, type)`: 参数 `type` 用于指定判别分析函数的类型，包括 `linear`（线性判别函数）、`diaglinear`（对角线性判别函数）、`quadratic`（二次判别函数）、

diagquadratic (对焦二次判别函数) 和 mahalanobis (分层方差估计的马氏距离法)。

- `class = classify(sample,training,group,type,prior)`: 参数 `prior` 用于各组的先验概率。
- `[class,err] = classify(...)`: 参数 `err` 返回判别分析的误差结果。
- `[class,err,posterior] = classify(...)`: 参数 `posterior` 返回后验概率的估计值。

【例 12.13】利用函数 `classify()` 进行判别分析。

现测定了 10 个已知点的大气污染状况, 其各污染气体的组分, 如表 12.6 所示, 根据已知的污染等级拟对测定的未知点的污染情况进行等级评定。

表 12.6 大气样品数据表

气体	氯	硫化氢	二氧化硫	环氧氯丙烷	环己烷	污染等级
已知样品 1	0.046	0.084	0.031	0.008	0.022	2
已知样品 2	0.043	0.045	0.100	0.022	0.017	2
已知样品 3	0.040	0.054	0.041	0.007	0.020	1
已知样品 4	0.025	0.050	0.130	0.025	0.026	2
已知样品 5	0.018	0.130	0.089	0.058	0.043	1
已知样品 6	0.040	0.120	0.070	0.050	0.016	3
已知样品 7	0.014	0.045	0.058	0.200	0.029	1
已知样品 8	0.020	0.050	0.068	0.220	0.039	1
已知样品 9	0.054	0.066	0.029	0.012	0.031	3
已知样品 10	0.055	0.036	0.019	0.010	0.040	3
未知样品 1	0.062	0.085	0.023	0.007	0.012	/
未知样品 2	0.042	0.075	0.120	0.021	0.007	
未知样品 3	0.035	0.112	0.052	0.016	0.011	

以下程序根据大气样品数据进行判别分析。

```
>> load classifydata
[class,err] = classify(sample,training,group)%利用函数 classify() 对大气样品数据进行判别分析
class =
     3
     2
     1
err =
0.111111111111111
```

12.5.2 聚类分析

聚类分析是研究样本分类问题的多元统计方法, 与判别分析不同的是事先不需要已知分类的数据训练建立判别准则, 而是直接根据数据特征建立分类规则, 对样本数据分类。在 MATLAB 中进行聚类分析的函数主要有如下几个。

1. pdist()函数

`pdist()`函数用于计算观测量间的成对距离, 其调用格式如下。

- `Y = pdist(X)`: 计算样本数据 `X` 的成对样本数据的距离, 数据矩阵 `X` 的每一行代表一个样本, 输出参数 `Y` 为样本数据中任意两个样本的距离。
- `Y = pdist(X,distance)`: 输入参数 `distance` 用于指定计算样本距离的函数, 具体参考 MATLAB 的帮助文档。
- `Y = pdist(X,distfun)`: 参数 `distfun` 为自定义的计算样本距离的函数。
- `Y = pdist(X,'minkowski',p)`: 以参数 `p` 的闵可夫斯基函数计算样本距离。

2. squareform()函数

利用函数 `pdist()`计算的样本数据为距离向量, 而函数 `squareform()`可用于距离向量和距离矩

阵的转换，其调用格式如下。

- $Z = \text{squareform}(y)$: 将 $\text{pdist}()$ 函数计算的距离向量 y 转换为平方距离 Z ，其中 $Z(i, j)$ 为样本 i 和 j 之间的距离。
- $y = \text{squareform}(Z)$: 将平方距离 Z 转换为向量距离 y 。

3. linkage()函数

函数 $\text{linkage}()$ 可用于创建系统聚类数，其调用格式如下。

- $Z = \text{linkage}(Y)$: 利用最短距离法创建系统聚类树 Z ， Y 为由函数 $\text{pdist}()$ 计算的向量距离。
- $Z = \text{linkage}(Y, 'method')$: 参数 $method$ 用于指定创建系统聚类树的方法，包括的方法有 $single$ (最短距离法，默认)、 $complete$ (最长距离法)、 $average$ (平均距离法)、 $weighted$ (加权平均法)、 $centroid$ (质心距离法)、 $median$ (加权质心距离法) 和 $ward$ (内平方距离法)。

4. dendrogram()函数

函数 $\text{dendrogram}()$ 可用于绘制树形聚类图，其调用格式如下。

- $H = \text{dendrogram}(Z)$: 输入参数 Z 为函数 $\text{linkage}()$ 计算所得的系统聚类树，绘制树形聚类图， H 为其中线条的句柄值。
- $H = \text{dendrogram}(Z, p)$: 参数 p 用于设置系统聚类树顶部的节点数，默认为 30。
- $[H, T] = \text{dendrogram}(...)$: 参数 T 为各样本观测对应的叶节点编号的列向量。
- $[H, T, perm] = \text{dendrogram}(...)$: 参数 $perm$ 返回树形图叶节点编号。
- $[...] = \text{dendrogram}(..., 'colorthreshold', t)$: 参数 $colorthreshold$ 根据阈值 t 设置树形图节点树的不同颜色。
- $[...] = \text{dendrogram}(..., 'orientation', 'orient')$: 控制聚类树形图的方向。
- $[...] = \text{dendrogram}(..., 'labels', S)$: 设置树形聚类图叶节点的标签。

5. cophenet()函数

函数 $\text{cophenet}()$ 可用于计算 Cophenetic 相关系数，其调用格式如下。

$c = \text{cophenet}(Z, Y)$: 输入参数 Z 为 $\text{linkage}()$ 函数创建的系统聚类树，输入参数 Y 为函数 $\text{pdist}()$ 的输出距离，返回 Cophenetic 相关系数。

6. cluster()函数

根据函数 $\text{linkage}()$ 计算所得的系统聚类树 Z 创建聚类，其调用格式如下。

- $T = \text{cluster}(Z, 'cutoff', c)$: 输入参数 Z 为系统聚类树矩阵， $cutoff$ 用于设置聚类的阈值，输出参数 T 返回聚类的结果，为每一个观测的聚类序号。
- $T = \text{cluster}(Z, 'maxclust', n)$: 参数 n 为最大聚类数。
- $T = \text{cluster}(..., 'criterion', 'crit')$: 参数 $criterion$ 用于设置聚类的标准，包括 $inconsistent$ (默认) 和 $distance$ 。
- $T = \text{cluster}(..., 'depth', d)$: 设置计算深度 d ，默认为 2。

7. clusterdata()函数

通过函数 $\text{clusterdata}()$ 可以直接进行聚类分析，在内部将调用上述的 $\text{pdist}()$ 、 $\text{linkage}()$ 和 $\text{cluster}()$ 函数，其具体的调用格式如下。

- $T = \text{clusterdata}(X, cutoff)$: 内部调用函数 $\text{pdist}()$ 、 $\text{linkage}()$ 和 $\text{cluster}()$ 进行聚类分析， $cutoff$ 用于设置聚类的阈值，返回聚类分析的结果 T 。

- `T = clusterdata(X,'param1',val1,'param2',val2,...)`: 设置聚类分析的相关参数, 包括 `distance` (聚类计算方法设置)、`linkage` (聚类方法设置)、`cutoff` (不一致系数或距离的阈值)、`maxclust` (最大分类数)、`criterion` (聚类的标准) 和 `depth` (计算深度)。

【例 12.14】 利用 MATLAB 进行聚类分析。

根据表 12.7 中的各大公司的评分数据进行聚类分析。

表 12.7 各大公司的评分数据

公司	组织文化	组织气氛	领导角色	员工发展
Microsoft	80	85	75	90
IBM	85	85	90	90
Dell	85	85	85	60
Apple	90	90	75	90
联想	99	98	78	80
NPP	88	89	89	90
北京电子	79	80	95	97
清华紫光	89	78	81	82
北大方正	75	78	95	96
TCL	60	65	85	88
娃哈哈	79	87	50	51
Angel	75	76	88	89
Hussar	60	56	89	90
世纪飞扬	100	100	85	84
Vinda	61	64	89	60

(1) 一步聚类。

```
>> T = clusterdata(x,'maxclust',3);
find(T==2) %显示分类的第二类
ans =
     1
     2
     3
     4
     5
     6
     7
     8
     9
    10
    12
    13
    14
```

(2) 分步聚类。

```
y=pdist(x);
z=linkage(y);
c=cophenet(z,y);
T=cluster(z,3);
H=dendrogram(z)
```

执行上述程序, 将生成如图 12.10 所示的聚类树形图。

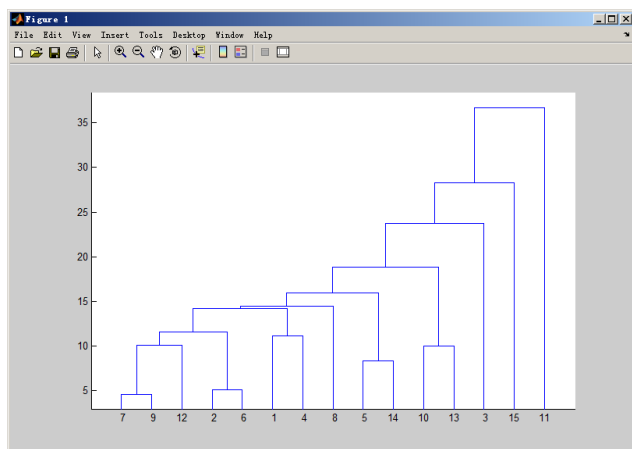


图 12.10 聚类树形图

12.5.3 主成分分析

主成分分析是一种常用的数据降维方法，可以从大量的数据中提取出几个关键主成分，这些主成分为原始变量的线性组合，且各主成分之间互不相关。在 MATLAB 中实现主成分分析的函数有如下几个。

1. pcacov()函数

函数 pcacov()可通过协方差矩阵进行主成分分析，其调用格式如下。

- $\text{COEFF} = \text{pcacov}(X)$: 输入参数 X 为协方差矩阵，返回参数为主成分系数 COEFF 。
- $[\text{COEFF}, \text{latent}] = \text{pcacov}(X)$: 返回参数 latent 为特征值向量。
- $[\text{COEFF}, \text{latent}, \text{explained}] = \text{pcacov}(X)$: 返回参数 explained 是主成分的贡献率向量。

2. princomp()函数

函数 princomp()可以对样本观测数据进行主成分分析，其调用格式如下。

- $\text{COEFF} = \text{princomp}(X)$: 对样本观测数据 X 进行主成分分析，其中 X 为 $n \times p$ 的数据矩阵，每行为一个样本观测，输出的主成分系数为 $p \times p$ 的矩阵。
- $[\text{COEFF}, \text{SCORE}, \text{latent}, \text{tsquare}] = \text{princomp}(X)$: 输出参数 SCORE 为主成分的得分，是 $n \times p$ 的数据矩阵，每行为一个观测，每列为一个主成分；输出参数 latent 为 p 个特征值构成的向量；输出参数 tsquare 为霍特林统计参量。当 $n \leq p$ (样本数小于等于变量数) 时， $\text{SCORE}(:, n:p)$ 和 $\text{latent}(n:p)$ 将为 0， $\text{COEFF}(:, n:p)$ 定义为与数据 X 正交。
- $[\dots] = \text{princomp}(X, 'econ')$: 参数 econ 用于控制 $n \leq p$ 时，不输出 $\text{SCORE}(:, n:p)$ 、 $\text{latent}(n:p)$ 和 $\text{COEFF}(:, n:p)$ 。

【例 12.15】利用 MATLAB 进行主成分分析。

(1) 通过协方差矩阵进行主成分分析。

```
>> load hald
X= cov(ingredients);
[COEFF, latent, explained] = pcacov(X)    %通过协方差矩阵进行主成分分析
COEFF =
Columns 1 through 3
-0.06779998569547    0.64601828656873   -0.56731454099051
-0.67851623541865    0.01999334048410    0.54396927658382
 0.02902083210623   -0.75530962249113   -0.40355346917267
 0.73087390945146    0.10848047717168    0.46839751838829
```



```

Column 4
0.50617955997771
0.49326809215930
0.51556741847684
0.48441622528920

```

```

latent =
1.0e+002 *
5.17796878073906
0.67496436048723
0.12405430048081
0.00237153265188

```

```

explained =
86.59738950184315
11.28823948155063
2.07470902838894
0.03966198821727

```

(2) 通过样本观测数据进行主成分分析。

```
>> [COEFF,SCORE,latent,tsquare] = princomp(ingredients)
```

%通过样本观测数据进行主成分分析

```
COEFF =
```

```
Columns 1 through 3
```

```

0.06779998569547    0.64601828656873   -0.56731454099051
0.67851623541865    0.01999334048410    0.54396927658382
-0.02902083210623   -0.75530962249113   -0.40355346917267
-0.73087390945146    0.10848047717168    0.46839751838829

```

```
Column 4
```

```

0.50617955997771
0.49326809215930
0.51556741847684
0.48441622528920

```

```
SCORE =
```

```
Columns 1 through 3
```

```

-36.82182599944971    6.87087815422737    4.59094445762975
-29.60727342071098   -4.61088196352630    2.24757816366395
12.98177571973762    4.20491318317594   -0.90224308269470
-23.71472572091802    6.63405255470872   -1.85474200080632
0.55319167662460     4.46173212317869    6.08741265232518
10.81249083330982     3.64657117454406   -0.91297079167460
32.58816660881793    -8.97984628493607    1.60626591399659
-22.60639549900560   -10.72590645736945   -3.23653771448342
9.26258723767583     -8.98537334747879    0.01690957810217
3.28396932964070     14.15727733750092   -7.04651299483377
-9.22003111782939    -12.38608078722045   -3.42834287828463
25.58490851742956     2.78169314815238    0.38671606686449
26.90316183467761     2.93097116504299    2.44552263019531

```

```
Column 4
```

```

0.39665258271391
-0.39584353669649
-1.12610058721062
-0.37856480838469
0.14238489604728
-0.13496881031468
0.08176392759995
0.32433477464637
-0.54374617598180
0.34050986096061

```

```

0.43515276966490
0.44681795054561
0.41160715640966
latent =
1.0e+002 *
5.17796878073906
0.67496436048723
0.12405430048081
0.00237153265188
tsquare =
5.68034084149095
3.07583703521197
6.00023279391262
2.61976323204795
3.36813944533665
0.56679667477121
3.48184073175696
3.97939790162016
2.60858624283357
7.48175632932763
4.18302223415259
2.23271872050565
2.72156781703209

```

12.5.4 因子分析

因子分析是指从观测变量中提取共性因子的多元统计分析技术，目前因子分析已经在很多领域得到了广泛的应用。在 MATLAB 中用于因子分析的函数为 `factoran()`，其调用格式如下。

- `lambda = factoran(X,m)`: 输入参数 `X` 为 $n \times d$ 的数据矩阵，行数据对应观测，列数据对应变量，`m` 为公因子数，返回参数 `lambda` 为因子模型的载荷矩阵。
- `[lambda,psi] = factoran(X,m)`: 参数 `psi` 返回特殊方差的最大似然估计值。
- `[lambda,psi,T] = factoran(X,m)`: 参数 `T` 为旋转矩阵。
- `[lambda,psi,T,stats] = factoran(X,m)`: 返回的结构体变量 `stats` 包含模型检验的信息，包括 `loglike` (对数似然函数的最大值)、`dfe` (误差自由度)、`chisq` (卡方统计量) 和 `p` (检验的概率值)。
- `[lambda,psi,T,stats,F] = factoran(X,m)`: 参数 `F` 返回因子得分矩阵。
- `[...] = factoran(...,'param1',value1,'param2',value2,...)`: 对因子分析算法的参数设置。

【例 12.16】 利用 MATLAB 进行因子分析。

```

load carbig
X = [Acceleration Displacement Horsepower MPG Weight];
X = X(all(~isnan(X),2),:);
[Lambda,Psi,T,stats] = factoran(X,2)    %因子分析
Lambda =
-0.24322160767381  -0.84998406009873
0.87734250290241   0.38708970362426
0.76182999477175   0.59295758318949
-0.79781813509127  -0.27860618990282
0.96924828067097   0.21293045868839
Psi =
0.21837034713875
0.08043169394902
0.06801636360408

```

```
0.28586481426728
0.01521839017913
T =
0.94758769427972    0.31949579285125
0.31949579285125   -0.94758769427972
stats =
    loglike: -0.00298161735157
         dfe: 1
        chisq: 1.15438285128113
         p: 0.28263354130857
```

12.6 本章小结

本章主要介绍了 MATLAB 的统计工具箱，包括了假设测验、方差分析、线性回归、非线性回归、多元统计等相关知识。通过本章的学习，读者将掌握如何利用 MATLAB 进行常规的统计分析操作，这些统计方法在各个领域都有很大的应用，可以有效地分析利用数据。



第13章

图像处理工具箱

图像处理是指利用计算机对图像进行恢复、校正、增强、分类和识别等处理，在气象、测绘、地理等很多领域都有广泛的应用。通过图像处理可以更好地认识了解图像反映的信息，从而利用计算机自动挖掘出有效的信息。

数字图像在 MATLAB 中是以一个或多个数据矩阵的形式存在的，因而利用 MATLAB 处理图像变得非常方便，同时 MATLAB 软件提供了专门的图像处理工具箱，内含丰富的图像处理函数。利用 MATLAB 图像处理工具箱可以快速而灵活地完成图像处理任务，用户可以直接调用图像处理的函数，完成包括图像读入和显示、获取图像信息、图像缩放/剪切、图像运算、图像增强、图像变换等基本的图像处理操作。

本章将具体讲述图像处理工具箱的使用。首先对 MATLAB 中支持的图像文件格式做简单的介绍，明确图像处理工具箱的处理对象。然后，介绍不同的图像类型及其转换，不同类型的图像在 MATLAB 中的数据存储形式有所差异，而不同的图像操作可能需要使用不同的图像类型文件。在明确了图像处理工具的处理对象的格式和类型后，我们将具体讲述图像处理工具箱如何完成基本的图像处理任务。

13.1 图像文件格式

目前的数字图像文件由于不同的使用目的、不同的图像要求、不同的图像压缩方式等因素往往在计算机中以不同的文件格式存在。MATLAB 图像处理工具箱功能强大，基本支持常用的图像格式文件的处理。其中，支持的图像文件格式主要有如下几种。

1. JPEG (Joint Photographic Experts Group) 格式

JPEG 是比较常见的一种图像文件格式，文件的扩展名为.jpg 或.jpeg。JPEG 图像文件使用有损压缩技术去除冗余的图像数据，所获取的图像虽然压缩率较高，文件的大小较小，但图像仍具有优良的品质。

2. BMP (Windows Bitmap) 格式

BMP 是 Windows 中的标准图像文件格式，使用非常广泛。BMP 图像文件格式采用位映射存储机制，而不使用图像压缩技术，有 1 位、4 位、8 位、24 位非压缩图像。BMP 格式的图像文件对 Windows 系统有较好的支持，且 BMP 图像文件是格式简单的位图图像，对图像文件的缩放操作效果更好，因而 BMP 图像文件格式也比较常用。

3. TIFF (Tagged Image File Format) 格式

TIFF 文件也是一种比较常用的图像文件格式，文件的扩展名为.tif 或.tiff，支持多种色彩位和色彩模式。TIFF 文件采用非失真的压缩模式，可以有效地去除图像文件中的重复信息，同时该格

式能较好地保持原图的颜色和层次，但是图像文件占用磁盘空间较大。

4. GIF (Graphics Interchange Format) 格式

GIF 图像文件采用可变长度等压缩算法对图像文件压缩，其压缩比较高，且磁盘空间占用较少，但是 GIF 最多支持 256 种色彩的图像。GIF 格式图像文件的另一个特点是在一个文件中可以同时存放多幅彩色图像，可以构建简单的动画文件，目前该格式的图像文件在网页中运用很多。

5. PCX (Windows Paintbrush) 格式

PCX 图像文件格式是最早支持彩色图像的文件格式之一，目前可以支持 256 种彩色。PCX 图像文件格式主要用于 Windows 系统中，具有强大的图像处理能力，也被很多图形图像软件工具所支持，也是目前比较常用的图像文件格式。

6. PNG (Portable Network Graphics) 格式

PNG 是一种比较新的图像文件格式，兼具 GIF 和 JPG 格式文件的优点，可以不失真的压缩图像，保持图像文件具有较好的品质。PNG 图像文件是 Photoshop 支持的主要图像文件格式，能提供无损压缩图像文件，但在其他一些绘图软件中的支持性可能不是很好。

13.2 图像类型及其转换

MATLAB 中支持真彩色图像、索引图像、灰度图像、二值图像 4 种不同的图像类型，不同类型的图像在 MATLAB 中的数据存储形式不同，因而在后面涉及的图像操作中也有所不同。本节主要介绍 MATLAB 不同的图像类型数据存储的差异，并介绍不同图像类型的转换，在实际应用中用户应该根据实际需要选择合适的图像类型。

13.2.1 真彩色图像 (RGB images)

真彩色图像是通过 R (红)、G (绿)、B (蓝) 3 个颜色分量的灰度值的组合来表示一个像素的颜色。对于像素大小为 $m \times n$ 的真彩色图像来说，在 MATLAB 中的数据存储结构为 $m \times n \times 3$ ，即 m 和 n 用于表示像素点的位置，而具体的颜色值通过像素点的 R、G、B 3 个分量的值确定，定义为 0 到 255 之间的数值。

【例 13.1】真彩色图像。

导入一幅 MATLAB 自带的真彩色图像，其中函数 `imread()` 可用于导入图像，在后续章节中将具体介绍，这里仅向读者演示导入 RGB 图像文件后在 MATLAB 工作空间中的存储结构。

```
I = imread('tissue.png');           %读入图像文件
R =                                 %显示图像文件 R 分量的值
    216
G =                                 %显示图像文件 G 分量的值
    221
B =                                 %显示图像文件 B 分量的值
    255
imshow(I)                           %真彩色图像的显示
```

执行上述程序，将生成如图 13.1 所示的真彩色图像。

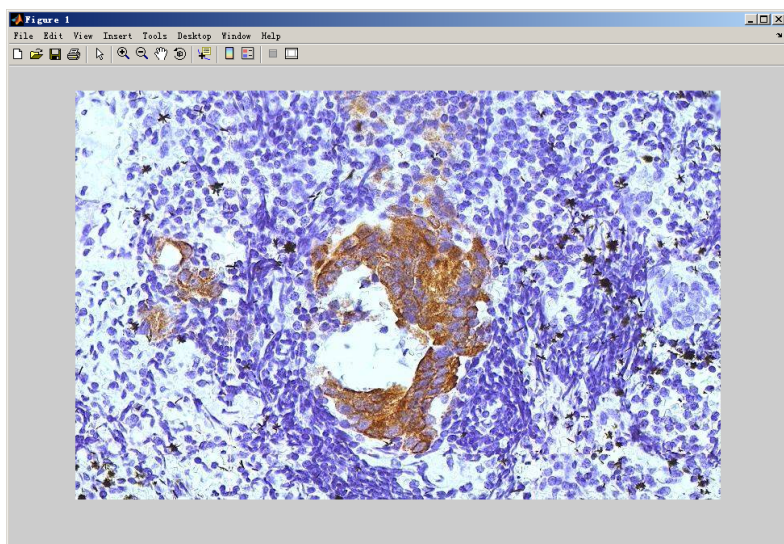


图 13.1 真彩色图像

在导入该图像后，读者可以观察到 MATLAB 工作空间内 `I` 变量的存储结构为 $506 \times 803 \times 3$ 的整型变量，其中 506 和 803 为图像的像素大小，而 3 代表 RGB3 个分量。

13.2.2 索引图像 (Index images)

索引图像包含数据索引矩阵和颜色映射矩阵两个数据结构。其中颜色映射矩阵是一个包含三列数据的矩阵，其中每一行对应一种颜色，每一行为 0 到 1 之间的三个浮点型数据，分别表示红、绿、蓝 3 种颜色的深度。

【例 13.2】索引图像。

```
load flujet;           %导入索引图像数据
imshow(X,map)          %索引图像的显示
```

执行上述程序，将生成如图 13.2 所示的索引图像。

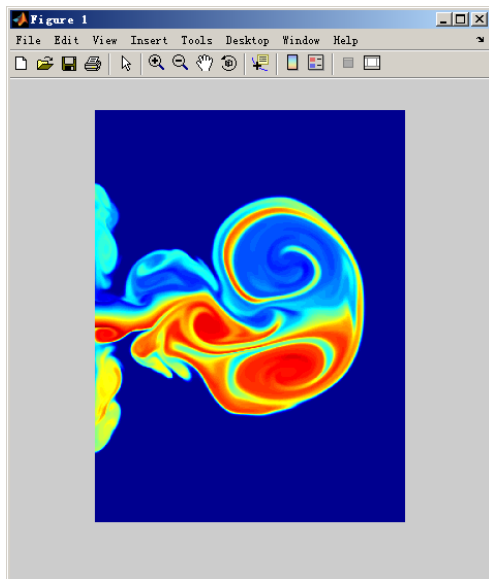


图 13.2 索引图像

在 MATLAB 工作空间中可以看到导入的图像包含索引数据矩阵 X 和对应的颜色映射矩阵 map ，数据矩阵 X 中每个元素为整型数据，对应颜色映射矩阵 map 相应行上的颜色，即当 $X(i, j)$ 的值为 m 时，图像 (i, j) 处的颜色为颜色映射矩阵 map 第 m 行对应的颜色。

13.2.3 灰度图像 (Intensity images)

存储灰度图像只使用一个数据矩阵存储图像，矩阵的每个元素为该像素点的灰度值，数据类型可以是整型或者浮点型。如果为双精度浮点型，则灰度图像的数据矩阵的范围为 $[0, 1]$ ；如果为 8 位无符号整型变量，则灰度图像的数据矩阵的范围为 $[0, 255]$ ；如果为 16 位无符号整型变量，则灰度图像的数据矩阵的范围为 $[0, 65535]$ 。

【例 13.3】灰度图像。

```
I = imread('testpat1.png');           %导入 MATLAB 内部的灰度图像
imshow(I)                             %显示灰度图像
```

执行上述程序，将生成如图 13.3 所示的灰度图像。

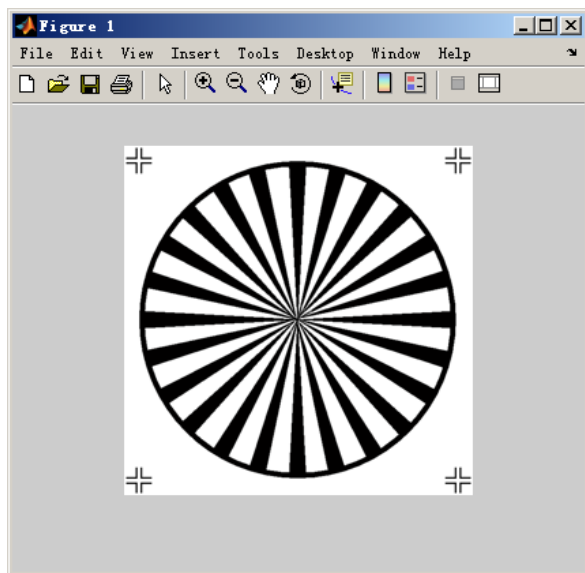


图 13.3 灰度图像

导入灰度图像后，可以看到在 MATLAB 工作空间中存储的数据 I 为 8 位无符号整型变量，图像每个像素点的数值为 $[0, 255]$ 之间的整数，用于表示像素的不同灰度值。

13.2.4 二值图像 (Binary images)

二值图像也只需一个数据矩阵即可完成图像的存储，其中每个像素只有 0 或 1 两个灰度值。二值图像图像的数据存储结构为逻辑变量，0 在图像中反映为白色，1 在图像中反映为黑色。

【例 13.4】二值图像。

```
BW = imread('circles.png');           %导入 MATLAB 中自带的二值图像
imshow(BW)                             %显示二值图像
```

运行上述程序后，将生成如图 13.4 所示的二值图像，其中的白色部分在存储二值图像的逻辑矩阵中对应的元素为 1，相应的黑色部分对应的元素为 0。

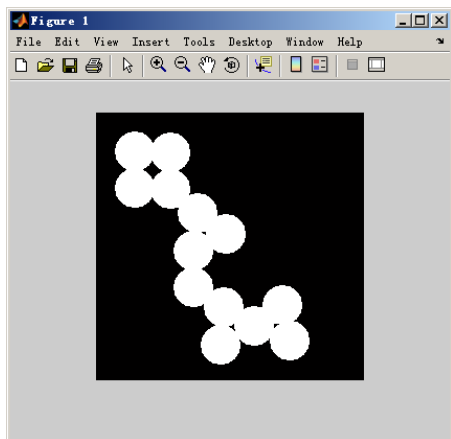


图 13.4 二值图像

13.2.5 图像类型转换

在实际的应用中由于不同的操作需要我们可能需要使用不同的图像类型，因而有必要实现不同类型图像的转换。在 MATLAB 中提供了以下 4 种图像类型相互转换的函数。

1. 函数 gray2ind()

函数 gray2ind() 用于将灰度图像或者二值图像转换为索引图像，其调用格式如下。

- `[X,map] = gray2ind(I,n)`: 使用颜色映射矩阵 `gray(n)` 将灰度图像 `I` 转换为索引图像 `X`，并返回其颜色矩阵 `map`，如果 `n` 默认为 64。
- `[X,map] = gray2ind(BW,n)`: 使用颜色映射矩阵 `gray(n)` 将二值图像 `BW` 转换为索引图像 `X`，并返回其颜色矩阵 `map`，如果 `n` 默认为 2。

【例 13.5】 利用函数 gray2ind() 将灰度图像转换为索引图像。

```
I = imread('testpat1.png');
[X,map] = gray2ind(I,32);           %灰度图像转换为索引图像
imshow(X, map);                     %显示转换后的索引图像
```

执行上述程序将生成如图 13.5 所示的转换后的索引图像。

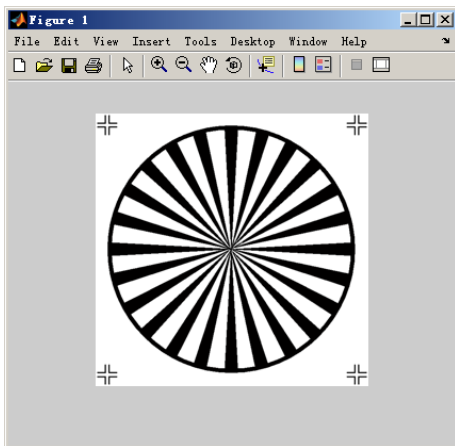


图 13.5 灰度图像转换为索引图像

2. 函数 dither()

函数 dither() 可用于通过抖动转换灰度图像到二值图像或者真彩色图像到索引图像。函数

dither()的调用格式如下。

- $X = \text{dither}(\text{RGB}, \text{map})$: 通过颜色抖动的方式将 RGB 图像转换为索引图像 X, map 为颜色映射矩阵, 不能超过 65 536 种颜色。
- $\text{BW} = \text{dither}(I)$: 通过颜色抖动的方式将灰度图像 I 转换为二值图像 BW。

【例 13.6】利用函数 dither()将灰度图像转换为二值图像。

```
I = imread('testpat1.png');           %将灰度图像转换为二值图像
BW = dither(I);                        %显示转换后的二值图像
imshow(BW)
```

执行上述程序, 将生成如图 13.6 所示的二值图像。

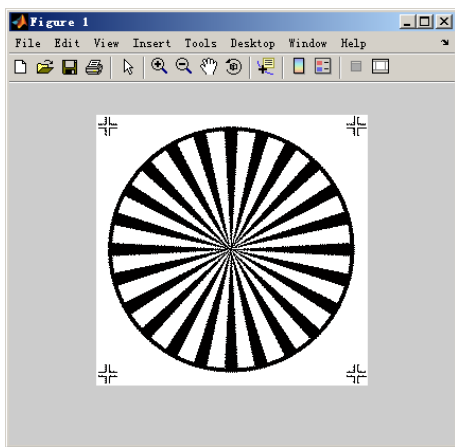


图 13.6 利用函数 dither()将灰度图像转换为二值图像

3. 函数 grayslice()

函数 grayslice()可用于将灰度图像转换为索引图像, 其基本的函数调用格式如下。

- $X = \text{grayslice}(I, n)$: 转换灰度图像 I 到索引图像 X, 通过阈值 $1/n, 2/n, \dots, (n-1)/n$ 。
- $X = \text{grayslice}(I, v)$: 转换灰度图像 I 到索引图像 X, 通过阈值 v, v 为 0 到 1 之间的向量。

【例 13.7】利用函数 grayslice()将灰度图像转换为索引图像。

```
I = imread('testpat1.png');           %读入灰度图像
X = grayslice(I, 32);                 %将灰度图像转换为索引图像
imshow(X, jet(32))                    %显示索引图像
```

执行上述程序, 将生成如图 13.7 所示的转换后的索引图像。

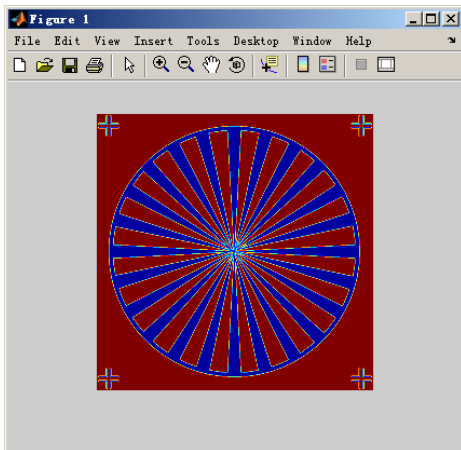


图 13.7 利用函数 grayslice()将灰度图像转换为索引图像

4. 函数 im2bw()

函数 im2bw() 可用于基于阈值将图像（灰度图像、索引图像、真彩色图像）转换为二值图像，函数在转换过程中先将不是灰度图像的图像转换为灰度图像，然后通过设定的阈值将灰度图像转换为二值图像，大于阈值的像素点输出值为 1，其他像素点的输出值为 0，设置的阈值范围为 0 到 1。函数 im2bw() 的调用格式如下。

- BW = im2bw(I,level): 基于阈值 level 转换灰度图像到二值图像。
- BW = im2bw(X,map,level): 基于阈值 level 转换索引图像到二值图像。
- BW = im2bw(RGB,level) 基于阈值 level 转换真彩色图像到二值图像。

【例 13.8】 利用函数 im2bw() 转换图像到二值图像。

```
I = imread('tissue.png');
subplot(1,2,1)
imshow(I)
title('RGB 图像')
x=im2bw(I,0.2);           %真彩色图像到二值图像的转换
subplot(1,2,2)
imshow(x)
title('二值图像')
```

执行上述程序，将生成如图 13.8 所示的图像，其中图像的左边为待转换的真彩色图像，图像的右边为转换后的二值图像。

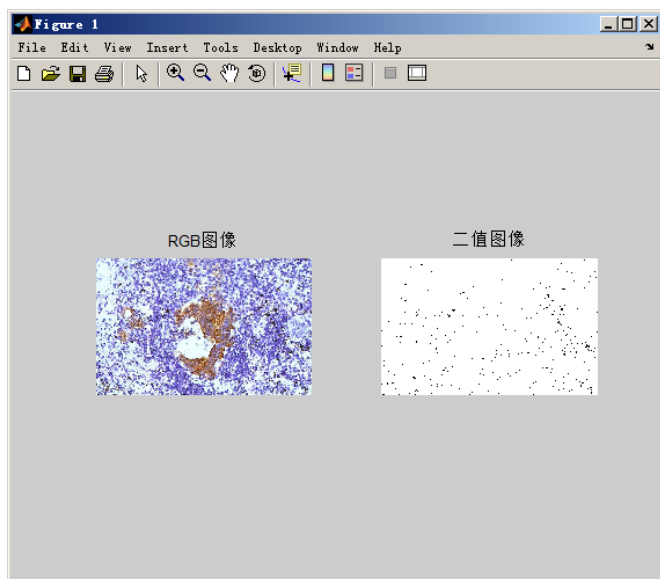


图 13.8 利用函数 im2bw() 转换图像到二值图像

5. 函数 ind2gray()

函数 ind2gray() 用于将索引图像转换为灰度图像，其调用格式如下。

I = ind2gray(X,map): 将索引图像 X 转换为灰度图像 I。

【例 13.9】 利用函数 ind2gray() 将索引图像转换为灰度图像。

```
load trees
I = ind2gray(X,map);       %转换索引图像到灰度图像
imshow(I)
```

执行上述程序，将生成如图 13.9 所示的灰度图像。

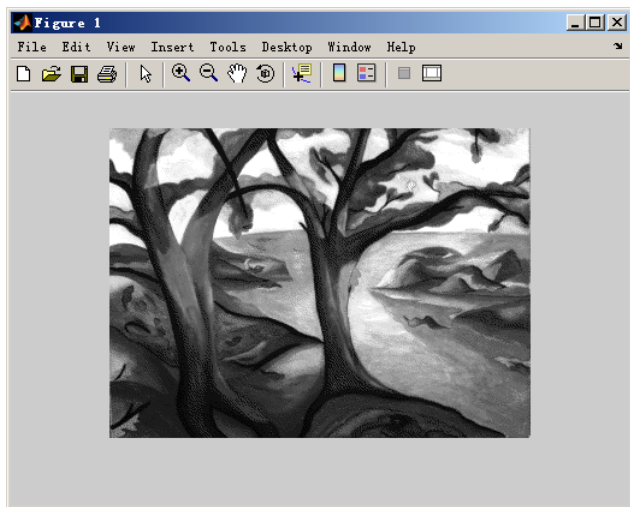


图 13.9 利用函数 `ind2gray()` 将索引图像转换为灰度图像

6. 函数 `ind2rgb()`

函数 `ind2rgb()` 用于将索引图像转换为真彩色图像，其调用格式如下。

`RGB = ind2rgb(X,map)`: 将索引图像 `X` 转换为真彩色图像 `RGB`。

【例 13.10】 利用函数 `ind2rgb()` 将索引图像转换为真彩色图像。

```
load trees
RGB = ind2rgb(X,map);           %将索引图像转换为真彩色图像
imshow(RGB)
```

执行上述程序，将生成如图 13.10 所示的真彩色图像。

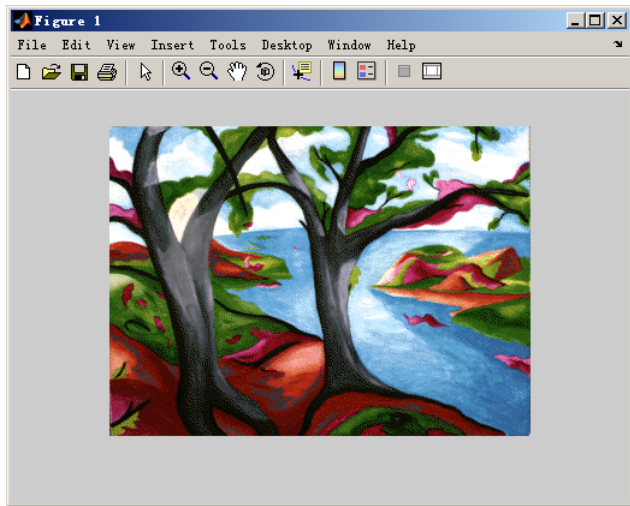


图 13.10 利用函数 `ind2rgb()` 将索引图像转换为真彩色图像

7. 函数 `mat2gray()`

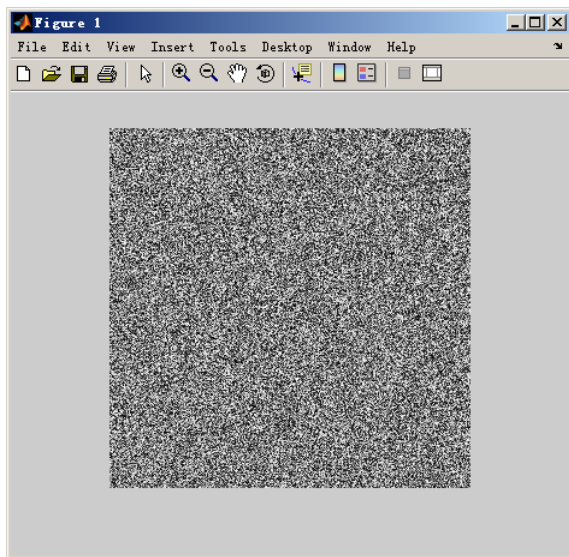
函数 `mat2gray()` 可用于将数据矩阵转换为灰度图像，其调用格式如下。

- `I = mat2gray(A,[amin amax])`: 转换数据矩阵 `A` 到灰度图像 `I`，其中 `amin` 对应灰度图像的 0 值（黑色），`amax` 对应灰度图像的 1 值（白色）。
- `I = mat2gray(A)`: 转换数据矩阵 `A` 到灰度图像 `I`，使用矩阵 `A` 的最小值对应灰度图像的 0 值（黑色），矩阵 `A` 的最大值对应灰度图像的 1 值（白色）。

【例 13.11】 利用函数 `mat2gray()` 可将数据矩阵转换为灰度图像。

```
A=rand(300,300);  
X=mat2gray(A);      %转换数据矩阵到灰度图像  
imshow(X)
```

执行上述程序将生成如图 13.11 所示的灰度图像。



13.11 利用函数 `mat2gray()` 可将数据矩阵转换为灰度图像

8. 函数 `rgb2gray()`

函数 `rgb2gray()` 可用于将真彩色图像转换为灰度图像，其调用格式如下。

`I = rgb2gray(RGB)`: 转换真彩色图像 RGB 到灰度图像 I。

【例 13.12】 利用函数 `rgb2gray()` 将真彩色图像转换为灰度图像。

```
RGB= imread('tissue.png');  
I = rgb2gray(RGB);      %转换真彩色图像为灰度图像  
imshow(I);
```

执行上述程序，将生成如图 13.12 所示的灰度图像。

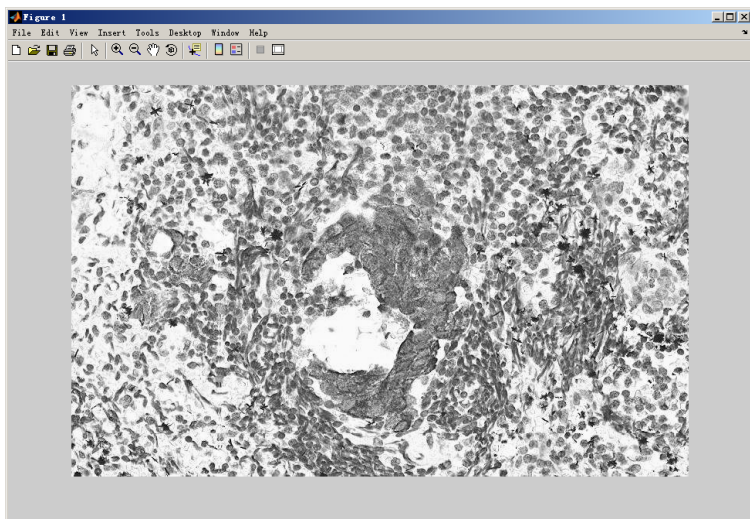


图 13.12 利用函数 `rgb2gray()` 将真彩色图像转换为灰度图像

9. 函数 rgb2gray()

函数 `rgb2gray()` 用于将真彩色图像转换为灰度图像，其调用格式如下。

- `[X,map] = rgb2ind(RGB,n)`: 将真彩色图像 RGB 转换为灰度图像 X，颜色映射矩阵 map 至少包含 n 种颜色，n 需小于 65536。
- `X = rgb2ind(RGB,map)`: 将真彩色图像 RGB 转换为灰度图像 X，通过颜色映射矩阵和 RGB 内颜色的最佳匹配，map 中的颜色数应少于 65536。
- `[X,map] = rgb2ind(RGB,tol)`: 使用统一的量化标准将真彩色图像 RGB 转换为灰度图像 X，颜色映射矩阵 map 至少包含 $(\text{floor}(1/\text{tol})+1)^3$ 种颜色，tol 的值必须在 0 到 1 之间。

【例 13.13】 利用函数 `rgb2gray()` 将真彩色图像转换为灰度图像。

```
RGB= imread('tissue.png');
[X,map]= rgb2ind(RGB,128);           %将真彩色图像转换为灰度图像
imshow(X,map)
```

执行上述程序，将生成如图 13.13 所示的图像。

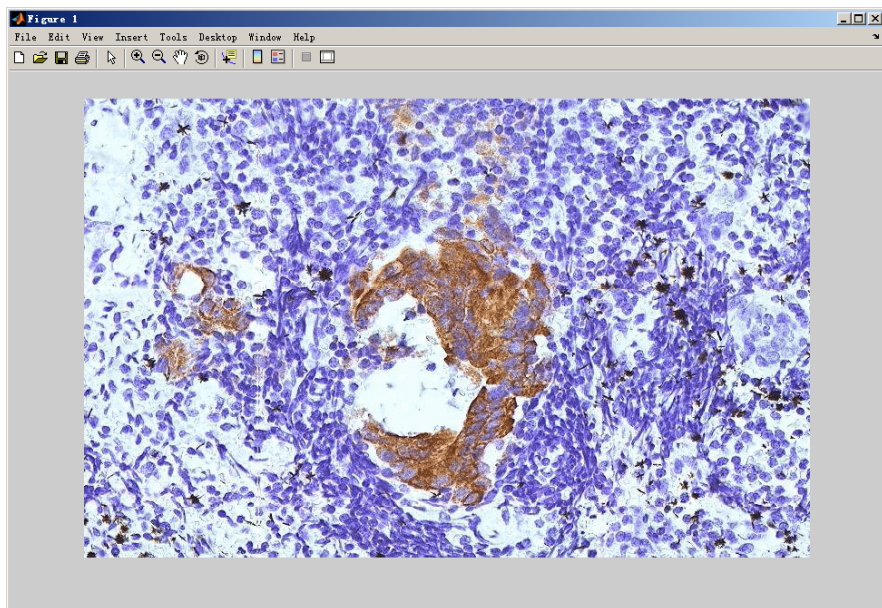


图 13.13 利用函数 `rgb2gray()` 将真彩色图像转换为灰度图像

13.3 图像处理的基本操作

在前面的章节中读者对 MATLAB 支持的图像的格式和类型有了基本的了解，在本节中将具体介绍如何对这些不同格式和类型的图像进行处理。

MATLAB 图像处理工具箱提供了丰富的图像处理操作功能，在此仅对图像处理的常用的常规操作做基本的介绍，包括图像读入和显示、获取图像信息、图像缩放/剪切、图像运算、图像增强、图像变换等基本的图像处理操作。

13.3.1 图像读入和显示

在处理一幅图像前首先需要了解图像的基本信息，并将其导入到 MATLAB 中，对于导入后的图像为了便于用户观察将显示出来。

1. 图像的信息查询

在 MATLAB 中提供了函数 `imfinfo()` 用于查询图像文件的信息，其调用格式如下。

- `info = imfinfo(filename,fmt)`: 导入文件名为 `filename` 的图像文件, 图像文件需要在 MATLAB 的当前路径或者搜索路径中, `fmt` 指定导入图像文件的格式, 当函数 `imfinfo()` 无法找到 `filename` 的图像文件时, 将搜索 `filename.fmt` 的图像文件, 返回的图像信息参数 `info` 包括 `Filename` (文件名)、`FileModDate` (文件最后的修改日期)、`FileSize` (文件大小)、`Format` (文件格式)、`FormatVersion` (文件版本)、`Width` (图形宽度, 单位为像素)、`Height` (图形高度, 单位为像素)、`BitDepth` (图像的深度) 和 `ColorType` (图像类型)。
- `info = imfinfo(filename)`: 查看图像文件 `filename` 的信息, 根据 `filename` 确定文件类型。

【例 13.14】图像信息查询。

```
>> info = imfinfo('tissue.png')           %查询图像信息
info =

    Filename: [1x46 char]
   FileModDate: '26-Jan-2003 08:03:08'
      FileSize: 1037034
        Format: 'png'
  FormatVersion: []
         Width: 800
        Height: 506
       BitDepth: 24
     ColorType: 'truecolor'
FormatSignature: [137 80 78 71 13 10 26 10]
      Colormap: []
     Histogram: []
InterlaceType: 'none'
Transparency: 'none'
SimpleTransparencyData: []
BackgroundColor: []
RenderingIntent: []
Chromaticities: []
         Gamma: []
   XResolution: []
   YResolution: []
ResolutionUnit: []
        XOffset: []
        YOffset: []
      OffsetUnit: []
SignificantBits: []
   ImageModTime: '17 Dec 2002 15:37:45 +0000'
         Title: []
        Author: []
   Description: [1x58 char]
   Copyright: [1x90 char]
  CreationTime: []
        Software: []
     Disclaimer: []
        Warning: []
         Source: []
        Comment: []
      OtherText: []
```

函数 `imformats()` 可用于查看 MATLAB 支持的图像文件的类型及其相关说明, 包括 `ext` (图像格式)、`isa` (图像格式的判断函数)、`info` (获取图像信息函数)、`read` (图像读入函数)、`write` (图像写回函数)、`alpha` (是否支持 alpha 通道) 和 `description` (图像格式描述)。

【例 13.15】利用函数 `imformats()` 查看不同格式图像文件的信息。

利用函数 `imformats()` 查看不同格式图像文件的信息, 在 MATLAB 命令窗口中输入 `imformats`,

在命令窗口将显示如下内容。

EXT	ISA	INFO	READ	WRITE	ALPHA	DESCRIPTION
-----	-----	-----	-----	-----	-----	-----
bmp	isbmp	imbmpinfo	readbmp	writebmp	0	Windows Bitmap (BMP)
cur	iscur	imcurinfo	readcur		1	Windows Cursor resources (CUR)
fts	fits	isfits	imfitsinfo	readfits		0 Flexible Image Transport System (FITS)
gif	isgif	imgifinfo	readgif	writегif	0	Graphics Interchange Format (GIF)
hdf	ishdf	imhdfinfo	readhdf	writehdf	0	Hierarchical Data Format (HDF)
ico	isico	imicoinfo	readico		1	Windows Icon resources (ICO)
jpg	jpeg	isjpg	imjpginfo	readjpg	writejpg	0 Joint Photographic Experts Group (JPEG)
pbm	ispbm	imnminfo	readpnm	writepnm	0	Portable Bitmap (PBM)
pcx	ispcx	impcxinfo	readpcx	writepcx	0	Windows Paintbrush (PCX)
pgm	ispgm	imnminfo	readpnm	writepnm	0	Portable Graymap (PGM)
png	ispng	impnginfo	readpng	writepng	1	Portable Network Graphics (PNG)
pnm	ispm	imnminfo	readpnm	writepnm	0	Portable Any Map (PNM)
ppm	isppm	imnminfo	readpnm	writepnm	0	Portable Pixmap (PPM)
ras	isras	imrasinfo	readras	writeras	1	Sun Raster (RAS)
tif	tiff	istif	imtifinfo	readtif	writetif	0 Tagged Image File Format (TIFF)
xwd	isxwd	imxwdinfo	readxwd	writexwd	0	X Window Dump (XWD)

2. 图像的读取

在 MATLAB 中函数 `imread()` 可用于导入图像文件，其调用格式如下。

- `A = imread(filename,fmt)`: `filename` 为导入图像的文件名，导入的图像文件名如果为相对路径需要在 MATLAB 的当前路径或者搜索路径下，或者导入的图像文件的文件名为绝对路径；`fmt` 为导入的图像文件的格式，如果函数找不到文件名为 `filename` 的图像文件，将继续查找 `filename.fmt`；导入后返回的参数为存储图像文件的数据矩阵，对于大小为 $m \times n$ 的图像文件，二值图像或者灰度图像导入后 `A` 为 $m \times n$ 的数据矩阵，而真彩色图像导入后的 `A` 为 $m \times n \times 3$ 的数据矩阵。
- `[X,map] = imread(filename,fmt)`: 此形式用于导入索引图像，其中返回的参数 `X` 为索引图像的数据矩阵，`map` 为对应的颜色映射矩阵。
- `[...] = imread(filename)`: 试图通过合适的格式导入文件名为 `filename` 的图像文件。
- `[...] = imread(URL,...)`: 导入网页 URL 中的图像文件。
- `[...] = imread(...,idx)`: 仅适用于 CUR、GIF、ICO 和 TIFF 格式的图像文件的导入。
- `[...] = imread(...,'PixelRegion',{ROWS, COLS})`: 仅适用于 TIFF 格式的图像文件的导入。
- `[...] = imread(...,'frames',idx)`: 仅适用于 GIF 格式的图像文件的导入。
- `[...] = imread(...,ref)`: 仅适用于 HDF 格式的图像文件的导入。
- `[...] = imread(...,'BackgroundColor',BG)`: 仅适用于 PNG 格式的图像文件的导入。
- `[A,map,alpha] = imread(...)`: 仅适用于 ICO、CUR 和 PNG 格式的图像文件的导入。

3. 图像的显示

对于利用函数 `imread()` 导入的图像文件，在 MATLAB 中仅以数字矩阵的形式存在，为了

便于观察图像，需要在窗口中显示图像。函数 `imshow()` 是最为常用的显示图像的函数，其调用格式如下。

- `imshow(I)`: 显示数据矩阵 `I` 的灰度图像。
- `imshow(I,[low high])`: 显示值域范围为 `[low high]` 的灰度图像。
- `imshow(IMG)`: 显示真彩色图像，`IMG` 为真彩色图像的数据矩阵。
- `imshow(BW)`: 显示二值图像，`BW` 为二值图像的数据矩阵。
- `imshow(X,map)`: 显示索引图像，`X` 为索引图像的数据矩阵，`map` 为索引图像对应的颜色映射矩阵。
- `imshow(filename)`: 直接显示文件名为 `filename` 的图像文件。
- `himage = imshow(...)`: 显示图像，并返回图形窗口句柄值 `himage`。
- `imshow(...,param1,val1,param2,val2)`: 显示图像并设置相关显示参数，可以设置的图像显示参数包括 `DisplayRange` (显示图像的范围设置)、`InitialMagnification` (显示图像的窗口范围设置)、`XData` (显示图像的 X 轴设置)、`YData` (显示图像的 Y 轴设置)。

【例 13.16】 图像文件的读取和显示。

```
load flujet; %导入索引图像数据矩阵
imshow(X,map) %显示索引图像
I = imread('testpat1.png'); %导入灰度图像
subplot(1,2,1)
imshow(I) %显示灰度图像
title('灰度图像的显示')
subplot(1,2,2)
imshow(I,[10 100]) %显示指定值域范围内的灰度图像
title('指定范围灰度图像的显示')
```

执行上述程序，将显示如图 13.14 所示的索引图像和图 13.15 所示的灰度图像。

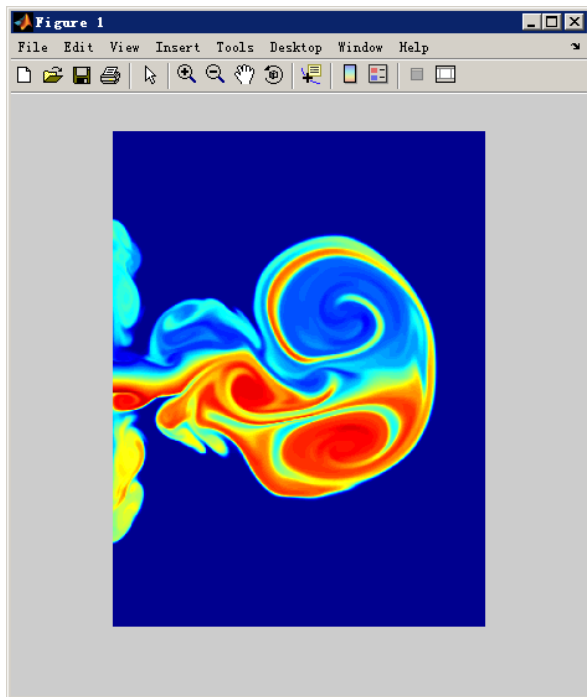


图 13.14 索引图像的显示

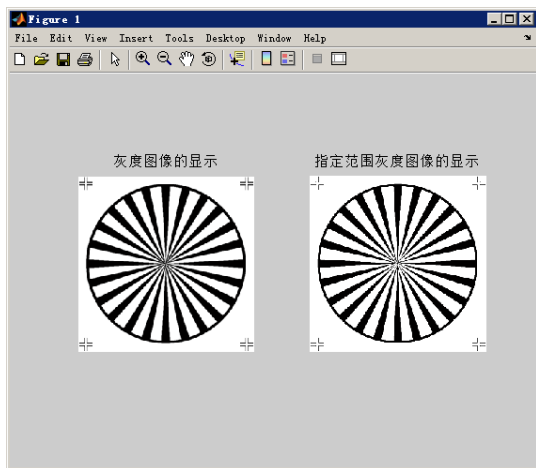


图 13.15 灰度图像的显示

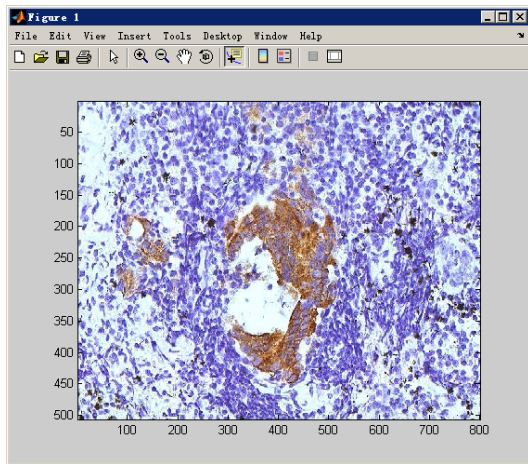
函数 `image()` 也可以用于图像的显示，与函数 `imshow()` 不同之处在于显示的图像带有坐标轴，可以方便用户查看像素大小。其调用格式如下。

- `image(C)`: 显示数据矩阵 `C` 代表的图像。
- `image(x,y,C)`: 显示图像，参数 `x` 和 `y` 分别设置坐标轴 `X` 轴和 `Y` 轴。

【例 13.17】 利用函数 `image()` 显示图像。

```
RGB= imread('tissue.png');
image (RGB)                %利用函数 image () 显示图像
```

执行上述程序，将显示如图 13.16 所示的图像。

图 13.16 利用函数 `image()` 显示图像

函数 `imagesc()` 用于可将当前图像的颜色映射到颜色矩阵的全范围，其调用格式如下。

- `imagesc(C)`: 显示数据矩阵 `C` 的图像，显示时数据矩阵 `C` 中的最小值对应于颜色映射表中的初始颜色值，数据矩阵 `C` 中的最大值对应于颜色映射表中的终值。
- `imagesc(x,y,C)`: 显示数据矩阵 `C` 的图像，并设置显示的坐标轴的范围。
- `imagesc(...,clims)`: 参数 `clims` 用于设置映射到颜色矩阵全范围的初值和终值，例如当前的颜色映射矩阵为 0 到 1 范围的灰度范围，则参数 `clims` 的第一个值对应灰度范围的起始值 0，`clims` 的第二个值对应灰度范围的终值 1。
- `h = imagesc(...)`: 显示图像并返回句柄 `h`。

【例 13.18】利用函数 `imagesc()` 显示图像。

```
I = imread('testpat1.png');    %读入图像
imagesc(I,[10 60])             %显示图像
colormap(gray)
```

执行上述程序将显示如图 13.17 所示的图像。

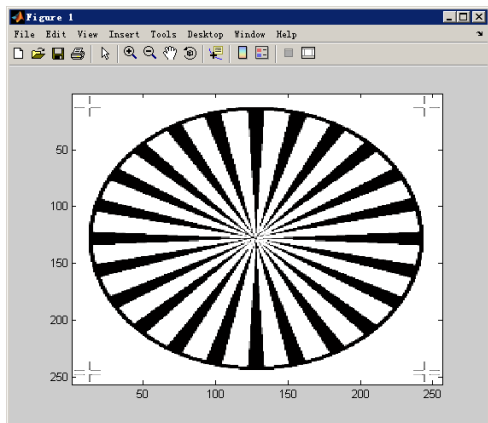


图 13.17 利用函数 `imagesc()` 显示图像

除了上面介绍的可用于显示图像的函数外，MATLAB 图像处理工具箱还提供了一些辅助函数用于帮助用户更好地显示图像，满足不同用户对图像显示的需要。下面具体介绍这些函数。

1) 函数 `colorbar()`

函数 `colorbar()` 可用于为显示的图像添加颜色条，便于用户查看不同颜色对应的数据值。函数 `colorbar()` 的调用格式如下。

- `colorbar`: 默认在图像的右侧添加颜色条。
- `colorbar('location')`: 添加颜色条，并设置添加的颜色条的位置。
- `cbar_axes = colorbar(...)`: 添加颜色条，并返回其句柄。

【例 13.19】颜色条的添加。

在图 13.17 的灰度图像上添加颜色条，在命令行输入：

```
colorbar
```

执行上述程序，将在图 13.17 的灰度图像上添加用于表示不同灰度颜色深度对应相应数值的颜色条，如图 13.18 所示。

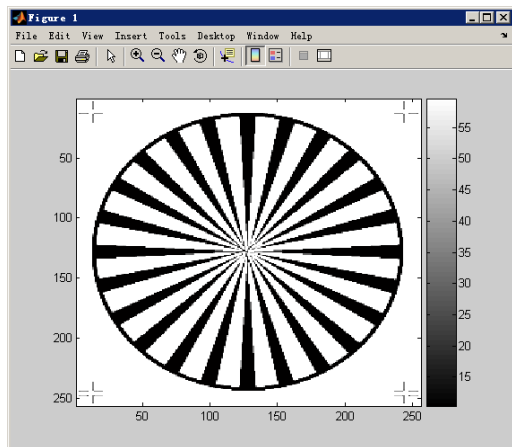


图 13.18 颜色条的添加

2) 函数 immovie()

函数 immovie() 可将多幅连续的图像制作成动画的形式, 其调用格式如下。

- mov = immovie(X,map): 用于连续索引图像的多帧连接, 连接的索引图像需要具有相同的大小。
- mov = immovie(RGB): 用于连续真彩色图像的多帧连接, 输出参数 RGB 需为 $m \times n \times 3 \times k$ 的数据矩阵, 其中 k 为连接的图像数。

【例 13.20】 多帧图像的动画显示。

```
load mri
mov = immovie(D,map);      %多帧索引图像的连接
movie(mov,5)               %以动画的显示连接的多帧索引图像
```

3) 函数 montage()

函数 montage() 可用于一次显示多幅图像, 每一幅图像将在图像窗口的不同位置显示。函数 montage() 的调用格式如下。

- montage(I): 多帧灰度图像的一次显示。
- montage(BW): 多帧二进制图像的一次显示。
- montage(X,map): 多帧索引图像的一次显示。
- montage(RGB): 多帧真彩色图像的一次显示。
- h = montage(...): 多帧图像的一次显示, 并返回其句柄 h。

4. 图像的导出

函数 imwrite() 用于图像文件的导出, 即将图像文件从 MATLAB 工作空间中保存到磁盘上。当然也可以向保存数据矩阵一样保存图像文件对应的数据, 而函数 imwrite() 主要以图像文件的形式向磁盘中写入不同格式的图像文件, 其调用格式如下。

- imwrite(A,filename,fmt): 其中参数 A 为导出图像的数据矩阵; filename 为导出图像的文件名, filename 可以为相对路径, 保存的图像文件在 MATLAB 当前路径下, 或者为指定的绝对路径; fmt 为导出图像的文件格式。
- imwrite(X,map,filename,fmt): 该格式用于导出索引图像。
- imwrite(...,filename): 导入图像文件, 文件格式通过文件名 filename 的扩展名确定。
- imwrite(...,Param1,Val1,Param2,Val2...): 导出图像文件并设置相关参数, 具体的参数参考帮助文档。

【例 13.21】 图像的导出。

```
A = imread('testpat1.png');
imwrite(A,'testpat2.png','PNG')      %图像写入硬盘
```

执行上述程序, 在 MATLAB 的当前搜索路径下将生成名为 testpat2.png 的图像文件。

13.3.2 图像缩放、旋转、剪切

本节主要介绍在 MATLAB 中如何实现对图像的一些几何操作, 包括图像缩放、旋转、剪切。下面具体介绍这些操作的函数实现。

1. 图像缩放

为了便于用户更好地观察、处理图像, 往往需要对图像进行一定程度的缩小和放大操作。对于图像的缩放操作会改变图像像素的大小, 其中会涉及图像的插值算法。

在 MATLAB 中图像缩放使用的插值方法包括最近邻插值、双线性插值法、双三次插值。其中, 最近邻插值法是最为简单常用的插值方法, 插值的图像的像素点的值为与其最邻近点的像素值, 该方法为 MATLAB 中图像缩放操作默认使用的方法; 双线性插值法插值的像素点为其领域范围 2×2 像素点的平均值; 双三次插值法是在插值像素点领域范围 4×4 像素点取平均值, 相对插值效

果优于双线性插值法,但算法的执行效率低。

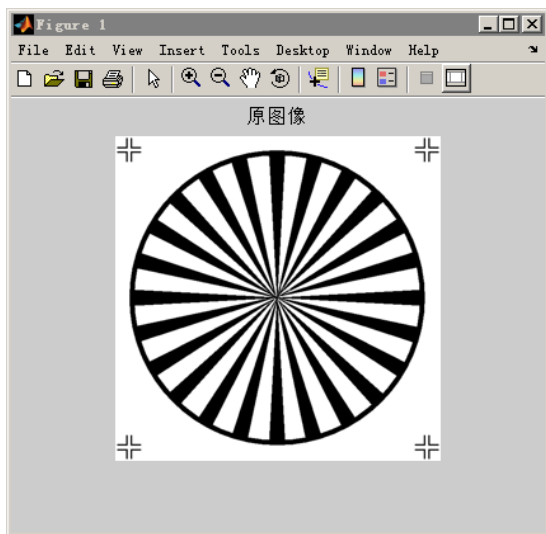
函数 `imresize()` 用于图像的缩放操作,其调用格式如下。

- `B = imresize(A,m)`: 参数 `A` 为待缩放的数据矩阵, `A` 可以为常用的各种图像类型(真彩色图像、灰度图像、索引图像、二值图像), `m` 为图像缩放的倍数,如果 `m` 为 0 到 1 之间的数将对图像 `A` 执行缩小操作,如果 `m` 为大于 1 的数,将对图像 `A` 执行放大操作,函数返回缩放后的图像 `B`,默认使用 `nearest`(最近邻插值法)对图像的缩放操作进行插值。
- `B = imresize(A,m,method)`: 对图像 `A` 执行缩放 `m` 倍的操作,并通过参数 `method` 指定缩放操作的插值方法,包括 `nearest`(最近邻插值法)、`bilinear`(双线性插值法)和 `bicubic`(双三次插值法)。
- `B = imresize(A,[mrows ncols],method)`: 通过参数 `mrows` 和 `ncols` 指定缩放后图像的行列像素大小,指定图像的缩放操作,该形式可不按原图像的长宽比对图像进行缩放。

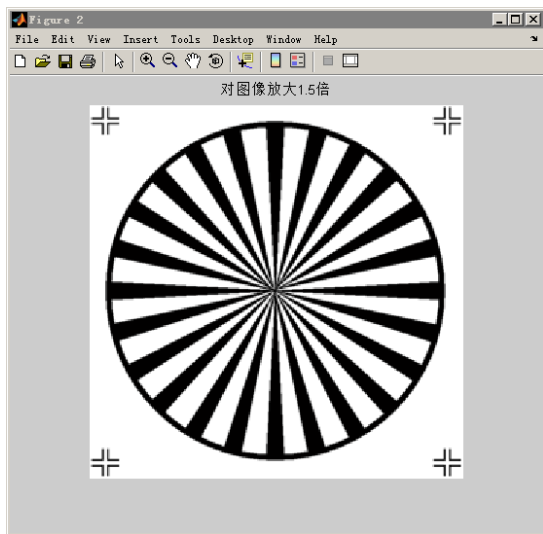
【例 13.22】 利用函数 `imresize()` 对图像执行缩放操作。

```
A = imread('testpat1.png');
imshow(A)
title('原图像')
B = imresize(A,1.5)           %对原图像放大 1.5 倍
figure
imshow(B)
title('对图像放大 1.5 倍')
```

执行上述程序,将生成如图 13.19 所示的缩放原图和放大 1.5 倍的图像。



(a) 原图



(b) 放大 1.5 倍后的图像

图 13.19 利用函数 `imresize()` 对图像执行缩放操作

2. 图像旋转

函数 `imrotate()` 可用于按一定角度旋转图像,其调用格式如下。

- `B = imrotate(A,angle)`: 图像 `A` 按照逆时针方向旋转角度 `angle`,如果需要按照顺时针方向旋转图像,则输入负值的旋转角度,旋转图像过程中使用的默认插值方法为最邻近插值法。
- `B = imrotate(A,angle,method)`: 旋转图像,并指定旋转过程中的插值算法,包括 `nearest`(最近邻插值法)、`bilinear`(双线性插值法)和 `bicubic`(双三次插值法),其中双三次插值法会产生超过原图像范围的像素点。

- `B = imrotate(A,angle,method,bbox)`: 参数 `bbox` 用于设置旋转过的图像的显示范围, 当取值为 `crop` 时, 原图像由于旋转操作部分图像被剪切, 仅显示与原图像一样大小的图像; 当取值为 `loose` 时将显示完整的放大的原图像, 显示的旋转后的图像尺寸大于原图像。

【例 13.23】 利用函数 `imrotate()` 旋转图像。

```
RGB= imread('tissue.png');
subplot(2,2,1);
imshow(RGB)
title('原图像')
%逆时针方向旋转 30 度
B=imrotate(RGB,30,'bilinear','crop');
subplot(2,2,2);
imshow(B);
title('crop 方式旋转过 30 度的图像')
%逆时针方向旋转 30 度
C=imrotate(RGB,30,'bilinear','loose');
subplot(2,2,3);
imshow(C);
title('loose 方式旋转过 30 度的图像')
%顺时针方向旋转 30 度
D = imrotate(RGB,-30);
subplot(2,2,4);
imshow(D)
title('顺时针旋转过 30 度的图像')
```

执行上述程序, 将生成如图 13.20 所示的旋转图像, 读者可以仔细观察各种调用格式图像的旋转效果。

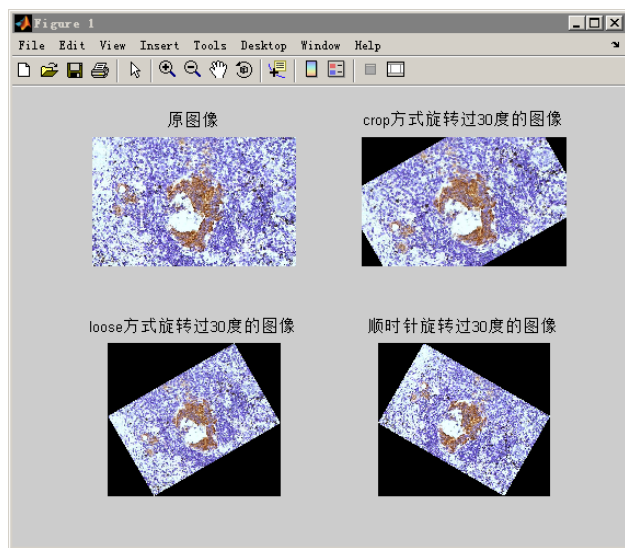


图 13.20 利用函数 `imrotate()` 旋转图像

3. 图像剪切

很多时候对于一幅大的图像用户可能只是关心其中的部分内容, 此时可以通过图像剪切操作把需要的图像剪切出来。在 MATLAB 中用于图像剪切操作的函数为 `imcrop()`, 其调用格式如下。

- `I2 = imcrop(I)`: 交互式的执行二进制、灰度、真彩色图像的剪切操作, 执行上述命令后将打开图像 `I` 的图像窗口, 用户在该窗口中通过鼠标执行选定区域的剪切操作, 剪切后的图像将返回到参数 `I2` 中。
- `X2 = imcrop(X,map)`: 交互式的对索引图像执行剪切操作。

- $I2 = \text{imcrop}(I, \text{rect})$: 对图像 I 执行剪切操作, 参数 rect 为向量 $[\text{xmin} \text{ymin} \text{width} \text{height}]$ 用于指定剪切图像的大小, 返回剪切后的图像 $I2$ 。
- $X2 = \text{imcrop}(X, \text{map}, \text{rect})$: 对索引图像执行指定区域的剪切操作。

【例 13.24】 利用函数 $\text{imcrop}()$ 剪切图像。

```
RGB= imread('tissue.png');
I = imcrop(RGB);           %手动剪切图像
imshow(I)
title('交互式的剪切后的图像')
I = imcrop(RGB,[100 100 250 150]); %在指定的区域剪切图像
figure
imshow(I)
title('指定区域剪切后的图像')
```

执行上述程序将首先弹出用户交互式的剪切图像的窗口, 剪切后的图像窗口如图 13.21 所示。同时函数 $\text{imcrop}()$ 用于剪切指定区域的图像, 如图 13.22 所示。

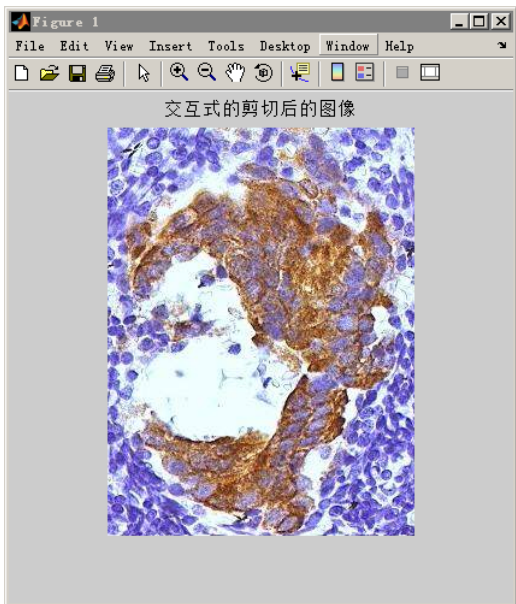


图 13.21 利用函数 $\text{imcrop}()$ 交互式的剪切图像

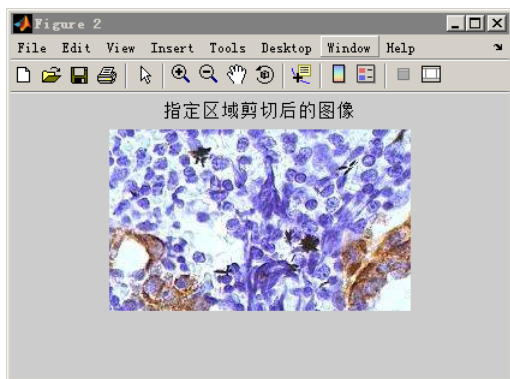


图 13.22 利用函数 $\text{imcrop}()$ 剪切指定区域的图像

13.3.3 图像的代数运算

本章主要介绍图像的代数运算, 包括图像的加、减、乘、除操作, 其本质是对图像文件在 MATLAB 中存储的数据矩阵执行代数运算操作, 但同时图像的代数运算具有一定的实际操作意义, 在显示图像后将可以达到不同的效果。下面具体介绍这 4 种图像的基本代数运算。

1. 图像的加法运算

图像的加法运算是指对图像数据矩阵的每个像素执行加法操作, 进行加法运算的图像文件需要具有相同的大小和图像类型。同时图像的加法运算还支持图像数据与某一固定的常数相加。在 MATLAB 中用于图像加法运算的函数为 $\text{imadd}()$, 其调用格式如下。

$Z = \text{imadd}(X, Y)$: 可以执行图像数据 X 和图像数据 Y 的加法运算, 此时 X 和 Y 需为相同大小、相同类型的图像文件, 也可以执行图像数据 X 和常数 Y 的加法运算, 此时图像文件 X 的每个像素都将执行和常数 Y 的相加运算, 返回的矩阵 Z 和 X 具有相同的大小。

【例 13.25】 利用函数 $\text{imadd}()$ 执行图像的加法运算。


```

I = imread('rice.png');
subplot(2,2,1)
imshow(I);
title('原图像 1')
J = imread('cameraman.tif');
subplot(2,2,2)
imshow(J);
title('原图像 2')
%执行两图像文件的相加操作，并显示图像
K = imadd(I,J);
subplot(2,2,3)
imshow(K)
title('两图像相加')
%执行图像文件和数据相加操作，并显示图像
L = imadd(I,50);
subplot(2,2,4)
imshow(L)
title('图像和常数相加')

```

执行上述程序，将显示如图 13.23 所示的图像，其中包含了原图像和图像相加后的图像。

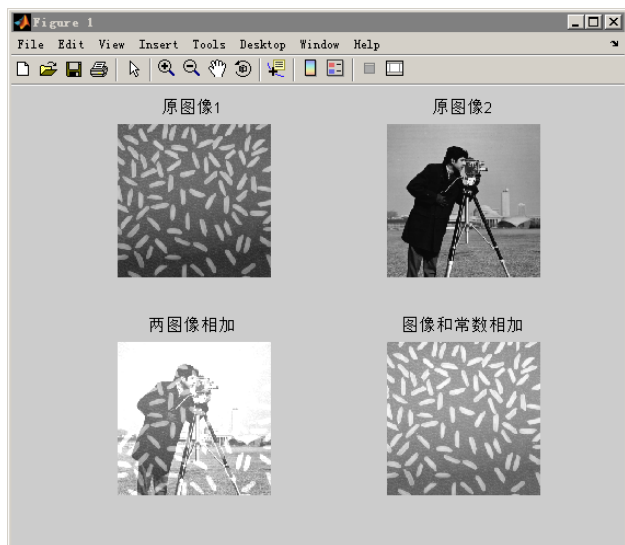


图 13.23 利用函数 imadd() 执行图像的加法运算

2. 图像的减法运算

图像的减法运算是两个图像数据矩阵的每一位的元素执行相应的减法运算，或者是对图像文件的数据矩阵所有元素都减去相同的一个数值。如果是图像之间相减，同样需要相减的图像数据具有相同的大小和数据类型，同时在减法的操作过程中可能相减后的数值小于 0，此时 MATLAB 将会对执行减法操作后，像素值小于 0 的点设置为 0。

在 MATLAB 用于图像减法运算的操作函数为 imsubtract()，其调用格式同函数 imadd()，在此不再详细展开叙述。

【例 13.26】利用函数 imsubtract() 执行图像的减法运算。

```

I = imread('rice.png');           %读入图像
subplot(2,2,1)
imshow(I);
title('原图像 1')
J = imread('cameraman.tif');       %读入图像
subplot(2,2,2)
imshow(J);

```

```

title('原图像 2')
K = imsubtract(I,J);           %图像的减法运算
subplot(2,2,3)
imshow(K)
title('原图像 1-原图像 2')
L = imsubtract(J,I);           %图像的减法运算
subplot(2,2,4)
imshow(L)
title('原图像 2-原图像 1')

```

执行上述程序，将生成如图 13.24 所示的图像的减法运算效果图。

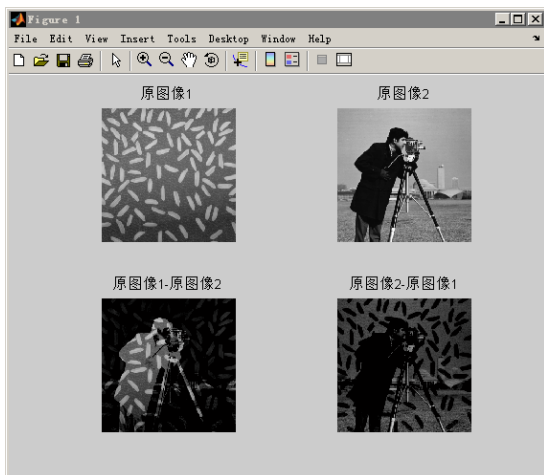


图 13.24 利用函数 imsubtract()执行图像的减法运算

3. 图像的乘法运算

在 MATLAB 中函数 immultiply()用于图像和图像或图像和单一数值的相乘运算，其调用格式与函数 imadd()和 imsubtract()类似，这里不再详细展开叙述。

【例 13.27】利用函数 immultiply()执行图像的乘法运算。

```

I = imread('rice.png');       %读入图像
L = immultiply(I,1.5);         %图像的乘法运算
imshow(L)                      %显示图像
title('图像和数值的相乘')

```

执行上述程序，将生成如图 13.25 所示的图像。

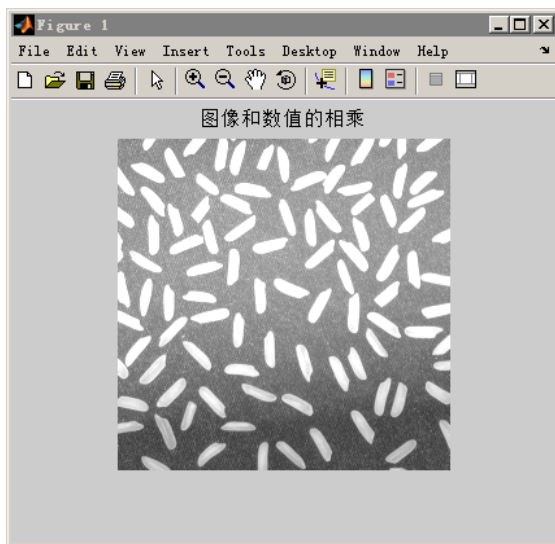


图 13.25 利用函数 immultiply()执行图像的乘法运算

4. 图像的除法运算

函数 imdivide()用于图像与图像或者图像与常数的除法运算，其使用方法也同之前介绍过的图像代数运算的各函数。

【例 13.28】利用函数 imdivide()执行图像的除法运算。

```
I = imread('rice.png');           %读入图像
subplot(2,2,1)
imshow(I);
title('原图像 1')
J = imread('cameraman.tif');      %读入图像
subplot(2,2,2)
imshow(J);
title('原图像 2')

K = imdivide(I,J);                %图像的除法运算
subplot(2,2,3)
imshow(K)
title('原图像 1/原图像 2')

L = imdivide(J,5);                %图像的除法运算
subplot(2,2,4)
imshow(L)
title('原图像 2/常数 5')
```

执行上述程序，将生成如图 13.26 所示的图像的除法运算图。

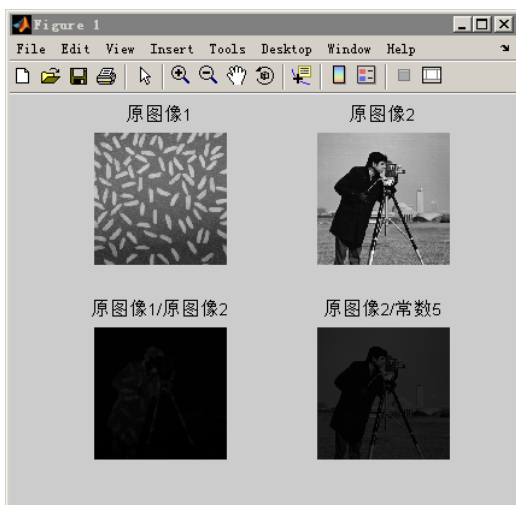


图 13.26 利用函数 imdivide()执行图像的除法运算

13.3.4 图像增强

为了更好地观察图像，使图像获得更好的视觉效果，便于用户获取更多的图像特征，我们可以对图像进行一定的增强处理。常用的图形增强操作包括对比度增强和直方图增强，下面具体介绍这些操作。

1. 对比度增强

在 MATLAB 图像处理工具箱中，`imadjust()`函数可通过线性变换调整图像的灰度范围，从而调整图像的对比度。函数 `imadjust()`的调用格式如下。

- `J = imadjust(I)`: 对图像 `I` 按照默认的方式进行线性变换，以增强图像的对比度，变换的方式为图像 `I` 的最低值对应于灰度范围的最低值，图像 `I` 的最高值对应于灰度范围的最高值。
- `J = imadjust(I,[low_in; high_in],[low_out; high_out])`: 线性变换图像 `I` 到图像 `J`，其对比度的变换方式为原图像 `I` 灰度范围 `[low_in; high_in]` 到变换后的图像 `J` 灰度范围 `[low_out; high_out]`，其中如果原图像中有小于 `low_in` 的值映射到 `low_out`，大于 `high_in` 的值映射到 `high_out`。
- `J = imadjust(...,gamma)`: 参数 `gamma` 为调节系数，如果 `gamma` 小于 1，图像变换时映射到较高的输出值；如果 `gamma` 大于 1，图像变换时映射到较低的输出值，默认状态下 `gamma` 值为 1 即线性映射。

【例 13.29】利用函数 `imadjust()`执行图像的线性变换。

```
I = imread('rice.png');           %读入图像
subplot(2,2,1)
imshow(I);
title('原图像')
J=imadjust(I);                     %全范围的图像增强
subplot(2,2,2)
imshow(J);
title('全范围的对比度增强')
K = imadjust(I,[0.3 0.5],[1]);    %局部范围的图像增强
subplot(2,2,3)
imshow(K);
title('局部范围的对比度增强')
gamma=1.5;
L = imdivide(J,gamma);
```

```
subplot(2,2,4)
imshow(L)
title('gamma=1.5')
```

执行上述程序将生成如图 13.27 所示的对比度增强变换的图像，读者可以仔细比较函数几种调用格式的差异。

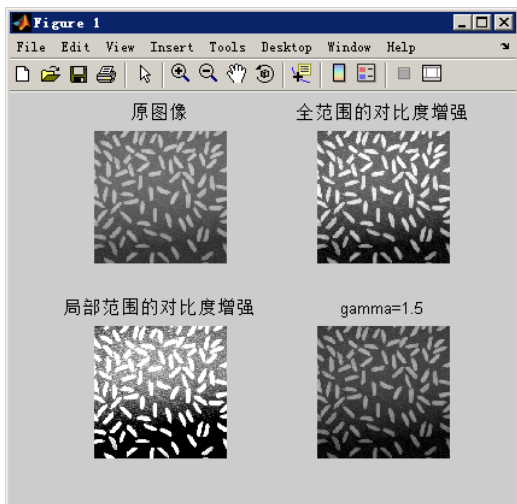


图 13.27 利用函数 `imadjust()` 执行图像的线性变换

2. 直方图增强

通过对图像的直方图调节以增强图像对比度是目前比较常用的图像增强操作。直方图用于反映图像像素点在灰度范围内的分布。在 MATLAB 中函数 `imhist()` 可用于计算和显示图像的直方图，而函数 `histeq()` 用于实现直方图的匹配和均衡化，以增强图形的对比度。下面具体介绍这两个函数的使用。

函数 `imhist()` 可用于计算并显示图像的直方图，其调用格式如下。

- `imhist(I,n)`: 计算并显示灰度图像或者二值图像的直方图，参数 `n` 用于设置直方图的类别数，如果图像 `I` 为二值图像，`n` 只能为 2。
- `imhist(X,map)`: 计算并显示索引图像 `X` 的直方图。

函数 `histeq()` 用于直方图的均衡化，直方图的均衡化是指将图像像素均匀地分布在图像的灰度值范围内。函数 `histeq()` 的调用格式如下。

- `J = histeq(I,hgram)`: 对图像 `I` 实行直方图均衡化操作，其中 `hgram` 为需要匹配的直方图，返回匹配后的图像 `J`。
- `J = histeq(I,n)`: 对图像 `I` 实行直方图均衡化操作，参数 `n` 为直方图均衡化后的灰度级数，默认情况下为 64。

【例 13.30】直方图增强。

```
I = imread('rice.png');      %读入图像
subplot(2,2,1)
imshow(I);
title('原图像')
subplot(2,2,2)
imhist(J);                  %直方图计算
title('显示原图像的直方图')
K = histeq(I);              %直方图增强
subplot(2,2,3)
imshow(K)
```

```

title('直方图增强后的图像')
subplot(2,2,4)
imhist(K);
title('直方图增强后图像的直方图')

```

执行上述程序，将生成如图 13.28 所示的图像。

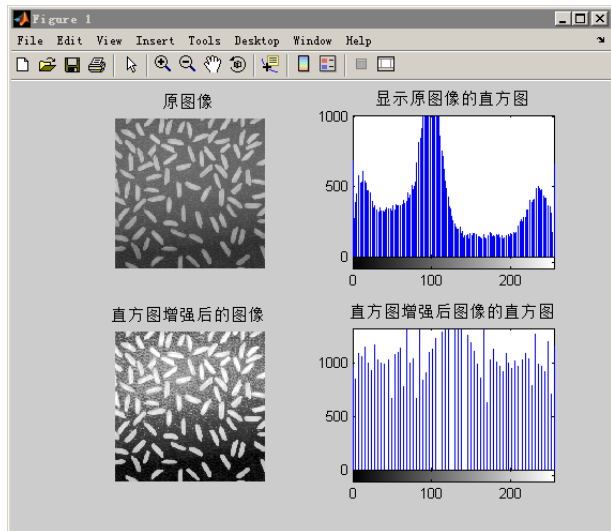


图 13.28 直方图增强

13.3.5 图像变换

图像变换是指把图像从原作用域变换到其他空间作用域，以便更好地处理分析图像。本小节主要介绍图像变换中比较常用的傅里叶变换和余弦变换的实现。

1. 傅里叶变换

在 MATLAB 图像处理工具箱中函数 `fft2()` 用于二维快速傅里叶变换，其调用格式如下。

- `Y = fft2(X)`: 对二维数据矩阵（图像数据矩阵）`X` 进行傅里叶变换，返回变换后的数据矩阵 `Y`，`X` 和 `Y` 具有相同的大小。
- `Y = fft2(X,m,n)`: 对二维数据矩阵 `X` 进行快速傅里叶变换，`m` 和 `n` 为返回的矩阵 `Y` 的大小，如果 `m` 和 `n` 超过数据矩阵 `X` 的大小，则在返回的 `Y` 矩阵处补 0。
- 函数 `ifft2()` 用于图像的二维傅里叶反变换，其调用格式如下。
- `Y = ifft2(X)`: 函数用于返回数据矩阵 `X` 的二维傅里叶反变换矩阵 `Y`，`X` 和 `Y` 具有相同大小。
- `Y = ifft2(X,m,n)`: 函数用于返回数据矩阵 `X` 的二维傅里叶反变换矩阵 `Y`，参数 `m` 和 `n` 确定返回的矩阵 `Y` 的大小。

函数 `fftshift()` 用于将傅里叶变换后的图像频谱中心从矩阵的原点移到矩阵的中心，便于观察变换效果，其调用格式如下。

`Y = fftshift(X)`: 移动数据矩阵 `X` 中的零频率成分从矩阵的原点到中心。

【例 13.31】 图像的快速傅里叶变换。

```

I = imread('rice.png');      %读入图像
subplot(2,2,1)
imshow(I);                   %显示原始图像
title('原图像')

J=fft2(I);                   %图像傅里叶变换
subplot(2,2,2)

```

```

imshow(J);
title('傅里叶变换后的图像')

K=fftshift(J);           %傅里叶变换后的图像频谱中心的移动
subplot(2,2,3)
imshow(K)
title('中心化后的图像')
L=ifft2(J)/255;          %傅里叶反变换
subplot(2,2,4)
imshow(L)
title('傅里叶反变换后的图像')

```

执行上述程序将生成如图 13.29 所示的快速傅里叶变换效果图。

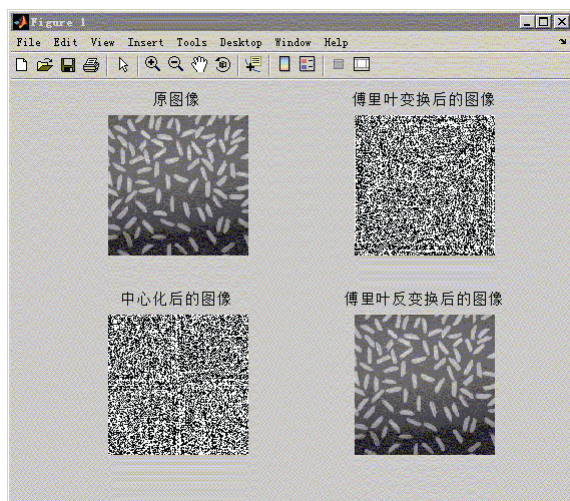


图 13.29 图像的快速傅里叶变换

2. 余弦变换

在 MATLAB 中函数 `dct2()` 和 `idct2()` 分别用于实现二维离散的余弦变换和反变换，其函数的调用格式如下。

- `B = dct2(A)`: 用于对图像数据矩阵 `A` 实现余弦变换，返回余弦变换后的数据矩阵 `B`。
- `B = dct2(A,m,n)`: 用于对图像数据矩阵 `A` 实现余弦变换，并指定返回的变换后的图像大小为 $m \times n$ 。
- `B = idct2(A)`: 对图像实现矩阵 `A` 的反余弦变换，变换后返回矩阵 `B`。
- `B = idct2(A,m,n)`: 对图像实现矩阵 `A` 的反余弦变换，变换后返回矩阵 `B`，大小为 $m \times n$ 。

【例 13.32】 图像的离散余弦变换。

```

I = imread('rice.png');           %读入图像
subplot(2,2,1)
imshow(I);                         %显示原图像
title('原图像')
J=dct(I);                          %余弦变换
subplot(2,2,2)
imshow(J);
title('余弦变换后的图像')
K=idct(J)/255;
subplot(2,2,3)
imshow(K)
title('反余弦变换后的图像')

```

执行上述程序将生成如图 13.30 所示的图像。

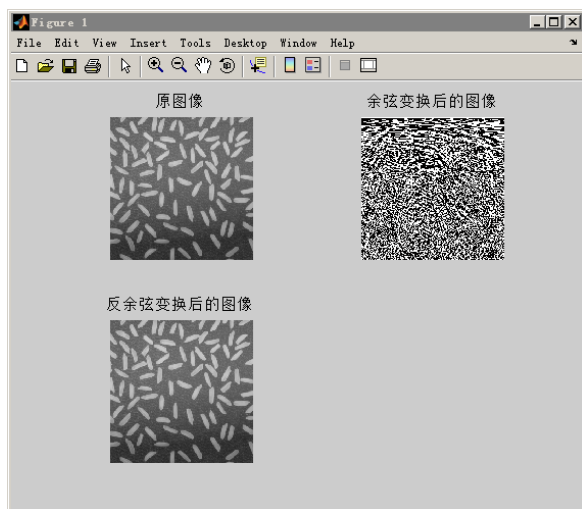


图 13.30 图像的离散余弦变换

13.3.6 图像滤波

图像滤波是常用的图像去噪、图像掩膜的手段,本小节将介绍主要的图像滤波方法在 MATLAB 中的实现,其中包括线性滤波和中值滤波等,下面具体介绍这些滤波方法。

1. 线性滤波

线性滤波是一种比较重要的滤波技术,线性滤波是通过像素点和其周围领域像素点简单的线性运算实现滤波的。在 MATLAB 中实现图像线性滤波的函数为 `imfilter()`,其调用格式如下。

`B = imfilter(A,H)`: 输入参数 `A` 为待滤波图像的数据矩阵, `H` 为线性滤波算子,返回滤波后的图像 `B`。

【例 13.33】 图像的线性滤波。

```
I = imread('rice.png');
H=ones(3)/5;
J = imfilter(I,H);           %图像的线性滤波
imshow(J)
```

执行上述程序,将生成如图 13.31 所示的图像。

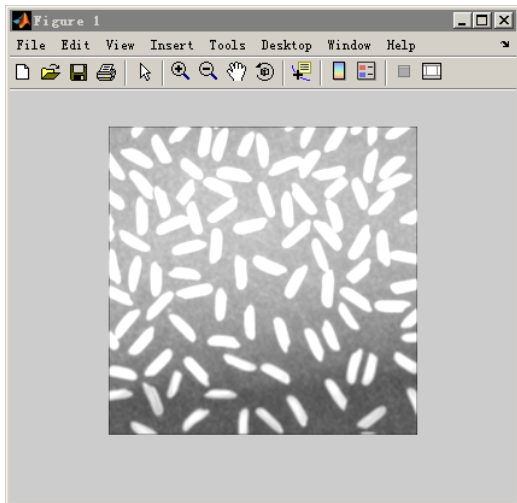


图 13.31 图像的线性滤波

在滤波过程中函数 `fspecial()` 可用于创建滤波算子，其调用格式如下。

- `h = fspecial(type)`: 参数 `type` 为设置滤波算子种类的参数，包括 `average` (均值滤波)、`gaussian` (高斯滤波)、`laplacian` (拉普拉斯)、`log` (拉普拉斯高斯滤波) 等 7 种常用的滤波算子的构建。
- `h = fspecial(type,parameters)`: 指定构建的滤波算子，并设置相应滤波算子的参数，具体详见帮助文档。

2. 中值滤波

中值滤波是一种非线性滤波技术，一定程度上可以抑制图像的“椒盐噪声”和“斑点噪声”。在 MATLAB 中用于中值滤波的函数为 `medfilt2()`。其调用格式如下。

- `B = medfilt2(A,[m n])`: `A` 为待滤波的图像的数据矩阵，`B` 为滤波后的数据矩阵，参数 `[m n]` 为中值滤波的邻域块的大小，默认为 3×3 。
- `B = medfilt2(A)`: 使用默认的邻域块对图像 `A` 进行中值滤波。

【例 13.34】 图像的线性滤波。

```
I = imread('rice.png');           %读入图像
subplot(2,2,1)
imshow(I);                         %显示图像
title('原图像')
J = imnoise(I,'salt & pepper',0.02); %图像加噪
subplot(2,2,2)
imshow(J);
title('含噪声的图像')
K = medfilt2(J);                  %中值滤波去噪
subplot(2,2,3)
imshow(K)
title('中值滤波后的图像')
```

执行上述程序将生成如图 13.32 所示的中值滤波的去噪效果图。

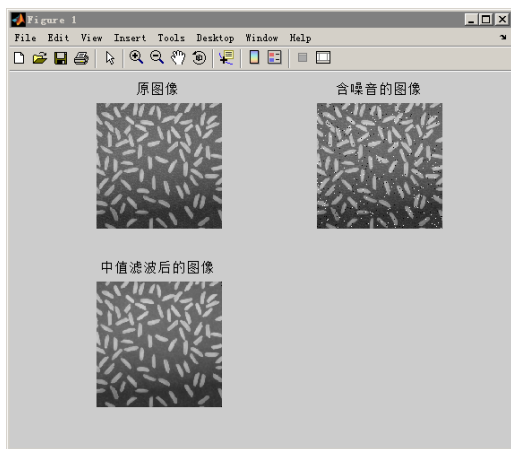


图 13.32 图像的中值滤波

13.4 本章小结

本章主要介绍了 MATLAB 的图像处理工具箱，主要包括以下几个方面：图像文件格式、图像类型及其转换、图像处理的基本操作。通过本章的学习读者将对图像处理的基础知识有所了解，并可利用 MATLAB 图像处理工具箱实现相应的图像处理操作。

第14章

优化工具箱

优化理论及其方法是求解许多工程领域实际问题的有效措施,而 MATLAB 软件提供的优化工具箱可以满足用户对优化计算的各种要求,近年来日益受到用户的欢迎。本章主要介绍 MATLAB 优化工具箱的使用,包括线性规划、整数规划、无约束规划和约束规划等常规的优化算法。

14.1 线性规划

线性规划问题是优化理论研究中比较成熟,且普遍使用的一种优化方法。当目标函数和约束条件都为线性的时候,即为线性规划问题。在 MATLAB 中求解线性规划问题的函数为 `linprog()`,可用于解决以下的优化问题:

$$\begin{aligned} \min_x & f^T x \\ \text{s.t.} & Ax \leq b, Aeqx = beq, lb \leq x \leq ub \end{aligned}$$

其函数的调用格式如下。

- `x = linprog(f,A,b,Aeq,beq)`: 输入参数 `f` 为目标函数,以目标函数的系数的列向量形式存在, `A` 和 `b` 为不等式约束条件的数据矩阵, `Aeq` 和 `beq` 为等式约束条件的数据矩阵; 函数返回的参数 `x` 为线性规划的最优解。
- `x = linprog(f,A,b,Aeq,beq,lb,ub)`: 参数 `lb` 和 `ub` 为线性优化问题中自变量的范围。
- `x = linprog(f,A,b,Aeq,beq,lb,ub,x0)`: 参数 `x0` 用于设置优化问题求解时解搜索的时候自变量的初始值。
- `x = linprog(f,A,b,Aeq,beq,lb,ub,x0,options)`: 参数 `options` 用于指定优化的参数,为结构体变量,通过函数 `optimset()` 设置。
- `[x,fval] = linprog(...)`: 返回的参数 `fval` 为最优解 `x` 处的函数值。
- `[x,fval,exitflag] = linprog(...)`: 返回的参数 `exitflag` 为解的输出标志,其值为 0 (优化过程中的迭代次数超过设定的最大迭代次数 `options.MaxIter.`)、1 (优化过程中找到最优解)、-2 (优化过程中没有找到可行解)、-3 (优化问题无解)、-4 (优化迭代的过程中遇到了空值 NaN) 和 -7 (搜索方向大小,优化迭代没有明显的变化)。
- `[x,fval,exitflag,output] = linprog(...)`: 输入参数 `output` 为包含优化信息的结构体变量,其中有优化的算法、迭代次数等信息。
- `[x,fval,exitflag,output,lambda] = linprog(...)`: 参数 `lambda` 为最优解 `x` 的拉格朗日乘子,包括 `lower` (下限值)、`upper` (上限值)、`ineqlin` (线性不等式) 和 `eqlin` (线性等式)。

【例 14.1】 线性规划问题的求解。

某厂利用甲、乙、丙 3 种原料生产 A、B、C 3 种产品,现根据生产每种产品在消耗甲、乙、

丙 3 种原料的消耗和利润, 以及甲、乙、丙 3 种原料可利用的数量, 如表 14.1 所示, 试制定最优的生产规划使得生产总利润最大。

表 14.1 某厂甲、乙、丙 3 种原料的消耗和利润

	生产单位产品消耗的原料			现有原料数
	甲	乙	丙	
A	5	3	2	700
B	3	5	2	500
C	2	3	1	800
单位产品利润	2	5	3	

根据题目要求, 需要求解的目标函数 $f = -2x_1 - 5x_2 - 3x_3$ 的最小值, 使得:

$$5x_1 + 3x_2 + 2x_3 \leq 700$$

$$3x_1 + 5x_2 + 2x_3 \leq 500$$

$$2x_1 + 3x_2 + x_3 \leq 800$$

$$x_1 \geq 0, x_2 \geq 0, x_3 \geq 0$$

利用函数 `linprog()` 求解该线性规划问题。

```
f = [-2; -5; -3]
A = [5 3 2
     3 5 2
     2 3 1];
b = [700; 500; 800];
lb = zeros(3,1);
>> [x,fval,exitflag,output,lambda] = linprog(f,A,b,[],[],lb) %线性规划问题求解
Optimization terminated.
x =
    1.0e+002 *
    0.000000000000000
    0.000000000000000
    2.500000000000000
fval =
   -7.499999999999999e+002
exitflag =
     1
output =
    iterations: 5
    algorithm: 'large-scale: interior point'
    cgiterations: 0
    message: 'Optimization terminated.'
lambda =
    ineqlin: [3x1 double]
    eqlin: [0x1 double]
    upper: [3x1 double]
    lower: [3x1 double]
```

在其中设置优化参数的函数 `optimset()` 也较为常用, 现具体介绍其调用格式如下。

- `options = optimset('param1',value1,'param2',value2,...)`: 用于对优化参数设置指定的值, 其中常用的参数包括 `Display` (控制计算过程中的显示, 取值为 `off` 时不显示输出, 取值为 `iter` 时每次迭代后显示输出的信息, 取值为 `final` 时显示最终输出结果, 默认, 取值为 `notify` 时当函数不收敛时显示输出结果)、`MaxIter` (函数优化过程中的最大迭代次数)、`TolFun` (算法终止的函数值精度) 和 `TolX` (算法终止的自变量的精度)。
- `optimset`: 该格式可用于查看完成的可以设置的算法参数的信息。
- `options = optimset`: 参数设置使用默认的设置。
- `options = optimset(optimfun)`: 返回优化函数 `optimfun` 的优化算法参数设置。
- `options = optimset(olddopts,'param1',value1,...)`: 使用老的算法设置 `olddopts`, 并更新其中

的部分参数。

- `options = optimset(oldopts,newopts)`: 使用新的算法设置 `newopts` 替代老的算法设置 `oldopts`。

14.2 整数规划

当需要求解的优化问题中全部或部分的变量为整数时为整数规划问题。在 MATLAB 优化工具箱中用于整数规划的函数为 `bintprog()`，主要解决的问题为：

$$\begin{aligned} \min_x & f^T x \\ \text{s.t.} & Ax \leq b, Aeqx = beg \end{aligned}$$

函数 `bintprog()` 的调用格式如下。

- `x = bintprog(f)`: 参数 `f` 为目标函数的系数矩阵。
- `x = bintprog(f, A, b)`: 参数 `A` 和 `b` 为整数规划的不等式约束条件。
- `x = bintprog(f, A, b, Aeq, beq)`: 参数 `Aeq` 和 `beq` 为整数规划的等式约束条件。
- `x = bintprog(f, A, b, Aeq, beq, x0)`: 参数 `x0` 为函数优化的初始值。
- `x = bintprog(f, A, b, Aeq, beq, x0, options)`: 参数 `options` 对优化过程进行一些参数设置。
- `[x, fval] = bintprog(...)`: 返回最优解 `x` 和最优解 `x` 处的函数值。
- `[x, fval, exitflag] = bintprog(...)`: 参数 `exitflag` 用于显示函数算法终止的状态，具体同函数 `linprog()`。
- `[x, fval, exitflag, output] = bintprog(...)`: 参数 `output` 返回优化算法的输出参数，包括 `iterations` (算法迭代的次数)、`nodes` (搜索到的节点数)、`time` (算法执行时间)、`algorithm` (使用的算法) 和 `message` (算法终止的原因)。

【例 14.2】 整数规划问题的求解。

$$\begin{aligned} \max z &= x_1 + 2x_2 - 3x_3 + x_4 \\ \text{s.t.} & \begin{cases} 2x_1 + 3x_2 + x_4 \geq 5 \\ x_1 - 2x_2 + 3x_3 - 2x_4 \geq -3 \\ 4x_1 - x_2 + 6x_3 - x_4 \geq 5 \\ x_i = 0 \text{ 或 } 1 \quad (i = 1, 2, 3, 4) \end{cases} \end{aligned}$$

首先对该问题进行转化，化为 MATLAB 整数规划函数可以解决的问题，其中，

求最大值 z 转换为计算： $\min z = -x_1 - 2x_2 + 3x_3 - x_4$

约束条件转换为：

$$\text{s.t.} \begin{cases} -2x_1 - 3x_2 - x_4 \leq -5 \\ -x_1 + 2x_2 - 3x_3 + 2x_4 \leq 3 \\ -4x_1 + x_2 - 6x_3 + x_4 \leq -5 \\ x_i = 0 \text{ 或 } 1 \quad (i = 1, 2, 3, 4) \end{cases}$$

通过函数 `bintprog()` 求解上述问题。

```
>> f=[-1 -2 3 -1]';
A=[-2 -3 0 -1;-1 2 -3 2;-4 1 -6 1];
b=[-5 3 -5]';
```

```
[x, fval, exitflag, output] = bintprog(f, A, b)
Optimization terminated.
x =
     1
     1
     1
     1
fval =
    -1
exitflag =
     1
output =
    iterations: 3
    nodes: 3
    time: 0.49920320000000
    algorithm: 'LP-based branch-and-bound'
    branchStrategy: 'maximum integer infeasibility'
    nodeSrchStrategy: 'best node search'
    message: 'Optimization terminated.'
```

14.3 二次规划

二次规划是一类特殊数学规划问题，在金融、工程等多领域都有很大的应用。函数 `quadprog()` 是 MATLAB 优化工具箱中专门用于二次规划问题的函数，主要解决的问题为：

$$\min_x \frac{1}{2} x^T H x + f^T x$$

$$\text{s.t. } A x \leq b; A_{eq} x = b_{eq}; lb \leq x \leq ub$$

其调用格式如下。

- `x = quadprog(H,f,A,b)`: 参数 `H` 为目标函数的海赛矩阵，`f` 为目标函数的一次项系数向量，参数 `A` 和 `b` 为二次规划的不等式约束条件。
- `x = quadprog(H,f,A,b,Aeq,beq)`: 参数 `Aeq` 和 `beq` 为二次规划的等式约束条件。
- `x = quadprog(H,f,A,b,Aeq,beq,lb,ub)`: 参数 `lb` 和 `ub` 为二次规划问题中自变量的范围。
- `x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0)`: 参数 `x0` 为函数优化的初始值。
- `x = quadprog(H,f,A,b,Aeq,beq,lb,ub,x0,options)`: 参数 `options` 对优化过程进行一些参数设置。
- `[x,fval] = quadprog(...)`: 返回最优解 `x` 和最优解 `x` 处的函数值。
- `[x,fval,exitflag] = quadprog(...)`: 参数 `exitflag` 用于显示函数算法终止的状态。
- `[x,fval,exitflag,output] = quadprog(...)`: 参数 `output` 为优化函数的输出，具体同函数 `linprog()`。
- `[x,fval,exitflag,output,lambda] = quadprog(...)`: 输出参数 `lambda` 同函数 `linprog()`。

【例 14.3】 二次规划问题的求解。

$$\min z = \frac{1}{2} x_1^2 + x_2^2 - x_1 x_2 - 2x_1 - 6x_2$$

$$\text{s.t. } \begin{cases} x_1 - x_2 \leq 2 \\ x_1 + x_2 \leq 5 \\ x_1 \geq 0, x_2 \geq 0 \end{cases}$$

```
H = [1 -1; -1 2];
f = [-2; -6];
```

```

A = [1 -1; 1 1];
b = [2; 5];
lb = zeros(2,1);
[x,fval,exitflag,output,lambda] = quadprog(H,f,A,b,[],[],lb) %根据目标函数和约束条件求解
二次规划问题
Optimization terminated.
x =
    2.200000000000000
    2.800000000000000
fval =
   -17.099999999999999
exitflag =
     1
output =
    iterations: 2
    algorithm: 'medium-scale: active-set'
 firstorderopt: []
   cgiterations: []
    message: 'Optimization terminated.'
lambda =
    lower: [2x1 double]
    upper: [2x1 double]
    eqlin: [0x1 double]
    ineqlin: [2x1 double]

```

14.4 非线性规划

非线性规划是求解具有非线性约束条件或非线性目标函数的数学规划。本节主要介绍 MATLAB 优化工具箱中求解无约束和约束非线性规划问题的函数。

14.4.1 无约束非线性规划

在 MATLAB 优化工具箱中提供了 3 个函数 (`fminbnd()`、`fminunc()`和 `fminsearch()`) 用于求解无约束的非线性规划问题。下面具体介绍这 3 个函数。

1. 函数 `fminbnd()`

函数 `fminbnd()`主要用于简单优化问题,可在固定的自变量区间内找到目标函数的最小值,其调用格式如下。

- `x = fminbnd(fun,x1,x2)`: 在自变量区间 $[x1, x2]$ 范围内找寻使目标函数 `fun` 的值最小的 `x`。
- `x = fminbnd(fun,x1,x2,options)`: 参数 `options` 用于设置优化算法的参数,可通过函数 `optimset()`设置。
- `[x,fval] = fminbnd(...)`: 返回最优解 `x` 和最优解 `x` 处的函数值。
- `[x,fval,exitflag] = fminbnd(...)`: 参数 `exitflag` 用于显示函数算法终止的状态。
- `[x,fval,exitflag,output] = fminbnd(...)`: 参数 `output` 为优化函数的输出,具体同函数 `linprog()`。

【例 14.4】利用函数 `fminbnd()`求解无约束非线性规划问题。

在当前路径下创建目标函数。

```

function y=fminbndfun(x)
y=x^2+3*x-5;

```

利用函数 `fminbnd()`求解目标函数在指定范围内的最小值。

```

>> [x,fval,exitflag,output] = fminbnd(@fminbndfun,-10,10)
x =
   -1.500000000000000

```

```
fval =
    -7.250000000000000
exitflag =
     1
output =
    iterations: 5
    funcCount: 6
    algorithm: [1x46 char]
    message: [1x112 char]
```

2. 函数 fminunc()

函数 fminunc() 可用于求解比较复杂的优化问题, 可计算一个无约束多元函数的最小值, 使用的算法包括 Large-Scale Algorithm (大规模算法) 和 Medium-Scale Algorithm (中等规模算法), 其调用格式如下。

- $x = \text{fminunc}(\text{fun}, x_0)$: 参数 fun 为优化的目标函数, x_0 为函数优化的初始值, 返回目标函数具有最小值的自变量 x 。
- $x = \text{fminunc}(\text{fun}, x_0, \text{options})$: 参数 options 用于设置优化算法的参数。
- $[x, \text{fval}] = \text{fminunc}(\dots)$: 返回最优解 x 和最优解 x 处的函数值。
- $[x, \text{fval}, \text{exitflag}] = \text{fminunc}(\dots)$: 参数 exitflag 用于显示函数算法终止的状态。
- $[x, \text{fval}, \text{exitflag}, \text{output}] = \text{fminunc}(\dots)$: 参数 output 为优化函数的输出, 具体同函数 linprog()。
- $[x, \text{fval}, \text{exitflag}, \text{output}, \text{grad}] = \text{fminunc}(\dots)$: 参数 grad 返回目标函数在最优解 x 处的梯度。
- $[x, \text{fval}, \text{exitflag}, \text{output}, \text{grad}, \text{hessian}] = \text{fminunc}(\dots)$: 参数 hessian 返回目标函数在最优解 x 处的 Hessian 矩阵值。

【例 14.5】 利用函数 fminunc() 求解无约束非线性规划问题。

计算 $\min z = x_1^2 + 2x_1x_2 + 3x_2^2$

在当前路径下创建目标函数。

```
function y=fminuncfun(x)
y = x(1)^2 + 2*x(1)*x(2) + 3*x(2)^2;
```

利用函数 fminunc() 求解目标函数在指定范围内的最小值。

```
>> x0=[1 1];
[x,fval,exitflag,output,grad,hessian] = fminunc(@fminuncfun,x0) %无约束非线性规划问题的求解
Optimization terminated: relative infinity-norm of gradient less than options.TolFun.

Computing finite-difference Hessian using user-supplied objective function.
x =
    1.0e-006 *
    -0.20293436685758    0.25408413857664
fval =
    1.317337980993761e-013
exitflag =
     1
output =
    iterations: 8
    funcCount: 27
    stepsize: 1
    firstorderopt: 1.163339581326216e-006
    algorithm: 'medium-scale: Quasi-Newton line search'
    message: [1x85 char]
grad =
    1.0e-005 *
    0.00873983822443
    0.11633395813262
hessian =
```

```
2.000000000000000    2.000000000000000
2.000000000000000    6.000000000000000
```

3. 函数 fminsearch()

函数 fminsearch() 多变量目标函数计算最小值，其调用格式如下。

- $x = \text{fminsearch}(\text{fun}, x_0)$: 参数 fun 为优化的目标函数， x_0 为函数优化解的初始值，返回目标函数具有最小值的自变量 x 。
- $x = \text{fminsearch}(\text{fun}, x_0, \text{options})$: 参数 options 用于设置优化算法的参数。
- $[x, \text{fval}] = \text{fminsearch}(\dots)$: 返回最优解 x 和最优解 x 处的函数值。
- $[x, \text{fval}, \text{exitflag}] = \text{fminsearch}(\dots)$: 参数 exitflag 用于显示函数算法终止的状态。
- $[x, \text{fval}, \text{exitflag}, \text{output}] = \text{fminsearch}(\dots)$: 参数 output 为优化函数的输出，具体同函数 linprog()。

【例 14.6】 利用函数 fminsearch() 求解无约束非线性规划问题。

计算 $\min z = (x_1^2 - 2x_1x_2)^2 + 3x_2^2$

在当前路径下创建目标函数。

```
function y=fminsearchfun(x)
y = (x(1)^2 - 2*x(1)*x(2)).^2 + 3*x(2)^2;
```

利用函数 fminsearch(), 求解目标函数在指定范围内的最小值。

```
>> x0=[1 1];
[x,fval,exitflag,output,grad,hessian] = fminunc(@fminsearchfun,x0) %无约束非线性规划问题的求解
Optimization terminated: relative infinity-norm of gradient less than options.TolFun.

Computing finite-difference Hessian using user-supplied objective function.
x =
    0.01138681963002    0.000000001880472
fval =
    1.681151777204311e-008
exitflag =
    1
output =
    iterations: 17
    funcCount: 54
    stepsize: 1
    firstorderopt: 5.905627039393124e-006
    algorithm: 'medium-scale: Quasi-Newton line search'
    message: [1x85 char]
grad =
    1.0e-005 *
    0.59056270393931
   -0.57480856936465
hessian =
    0.00158947918072   -0.00156147246158
   -0.00156147246158    6.00103727729029
```

14.4.2 约束非线性规划

在 MATLAB 优化工具箱中提供了两个函数 (fgoalattain() 和 fminimax()) 用于求解约束的非线性规划问题。下面具体介绍这两个函数。

1. 函数 fgoalattain()

函数 fgoalattain() 可用于求解多目标规划问题，其调用格式如下。

- $x = \text{fgoalattain}(\text{fun}, x_0, \text{goal}, \text{weight})$: 输入参数 fun 为目标函数， x_0 为函数优化解的初始值，goal 为目标函数指定的目标，参数 weight 为指定的权重，返回目标函数具有最小值的自变

量 x_0 。

- $x = \text{fgoalattain}(\text{fun}, x_0, \text{goal}, \text{weight}, A, b)$: 参数 A 和 b 为线性不等式的约束条件, $A*x \leq b$ 。
- $x = \text{fgoalattain}(\text{fun}, x_0, \text{goal}, \text{weight}, A, b, Aeq, beq)$: 参数 Aeq 和 beq 为等式约束条件, $Aeq*x = beq$ 。
- $x = \text{fgoalattain}(\text{fun}, x_0, \text{goal}, \text{weight}, A, b, Aeq, beq, lb, ub)$: 参数 lb 和 ub 用于限制自变量的范围, $lb \leq x \leq ub$ 。
- $x = \text{fgoalattain}(\text{fun}, x_0, \text{goal}, \text{weight}, A, b, Aeq, beq, lb, ub, \text{nonlcon})$: 参数 nonlcon 用于定义非线性不等式约束条件。
- $x = \text{fgoalattain}(\text{fun}, x_0, \text{goal}, \text{weight}, A, b, Aeq, beq, lb, ub, \text{nonlcon}, \text{options})$: 参数 options 用于设置算法的优化参数。
- $[x, \text{fval}] = \text{fgoalattain}(\dots)$: 返回最优解 x 和最优解 x 处的函数值。
- $[x, \text{fval}, \text{attainfactor}] = \text{fgoalattain}(\dots)$: attainfactor 是指函数目标达到情况, 当 attainfactor 值为正时, 目标达到最优解 fval 时没有溢出目标 goal , 当 attainfactor 为负时, 最优解 fval 有溢出目标 goal 的情况。
- $[x, \text{fval}, \text{attainfactor}, \text{exitflag}] = \text{fgoalattain}(\dots)$: 参数 exitflag 用于显示函数算法终止的状态。
- $[x, \text{fval}, \text{attainfactor}, \text{exitflag}, \text{output}] = \text{fgoalattain}(\dots)$: 参数 output 为优化函数的输出, 具体同函数 $\text{linprog}()$ 。
- $[x, \text{fval}, \text{attainfactor}, \text{exitflag}, \text{output}, \text{lambda}] = \text{fgoalattain}(\dots)$: 参数 lambda 为最优解 x 的拉格朗日乘子。

【例 14.7】 利用函数 $\text{fgoalattain}()$ 求解约束非线性规划问题。

求解多目标问题:

$$\begin{aligned} \min z_1 &= 21x_1 + 48x_2 \\ \max z_2 &= 36x_1 + 65x_2 \\ \text{s.t.} \quad &\begin{cases} x_1 \leq 5 \\ x_2 \leq 8 \\ x_1 + x_2 \geq 10 \\ x_1 \geq 0; x_2 \geq 0 \end{cases} \end{aligned}$$

在当前路径下创建目标函数:

```
function f=fgoalattainfun(x)
f(1)=21*x(1)+48*x(2);
f(2)=-36*x(1)-65*x(2);
```

利用函数 $\text{fgoalattain}()$ 求解目标函数在指定范围内的最小值。

```
>> goal=[30,-45];
weight=[30,-45];
x0=[2,2];
A=[1 0; 0 1;-1 -1];
b=[5,8,-10];
lb=zeros(2,1);
[x,fval,attainfactor,exitflag]=fgoalattain(@fgoalattainfun,x0,goal,weight,A,b,[],[],lb,[],)%约束非线性规划问题的求解
Optimization terminated: magnitude of search direction less than 2*options.TolX
and maximum constraint violation is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
    lower    upper    ineqlin    ineqnonlin
         1         1
         3
x =
```

```

5      5
fval =
345 -505
attainfactor =
10.500000000000000
exitflag =
4

```

2. 函数 fminimax()

函数 fminimax() 可用于求解有约束的最小值和最大值问题，其调用格式如下。

- $x = \text{fminimax}(\text{fun}, x_0)$: 从初始值 x_0 开始寻找目标函数 fun 的最小值，返回目标函数具有最小值的自变量 x_0 。
- $x = \text{fminimax}(\text{fun}, x_0, A, b)$: 参数 A 和 b 为线性不等式的约束条件， $A*x \leq b$ 。
- $x = \text{fminimax}(\text{fun}, x_0, A, b, Aeq, beq)$: 参数 Aeq 和 beq 为等式约束条件， $Aeq*x = beq$ 。
- $x = \text{fminimax}(\text{fun}, x_0, A, b, Aeq, beq, lb, ub)$: 参数 lb 和 ub 用于限制自变量的范围， $lb \leq x \leq ub$ 。
- $x = \text{fminimax}(\text{fun}, x_0, A, b, Aeq, beq, lb, ub, \text{nonlcon})$: 参数 nonlcon 用于定义非线性不等式约束条件。
- $x = \text{fminimax}(\text{fun}, x_0, A, b, Aeq, beq, lb, ub, \text{nonlcon}, \text{options})$: 参数 options 用于设置算法的优化参数。
- $[x, fval] = \text{fminimax}(\dots)$: 返回最优解 x 和最优解 x 处的函数值。
- $[x, fval, \text{maxfval}] = \text{fminimax}(\dots)$: 参数 maxfval 返回目标函数的最大值。
- $[x, fval, \text{maxfval}, \text{exitflag}] = \text{fminimax}(\dots)$: 参数 exitflag 用于显示函数算法终止的状态。
- $[x, fval, \text{maxfval}, \text{exitflag}, \text{output}] = \text{fminimax}(\dots)$: 参数 output 为优化函数的输出。
- $[x, fval, \text{maxfval}, \text{exitflag}, \text{output}, \text{lambda}] = \text{fminimax}(\dots)$: 参数 lambda 为最优解 x 的拉格朗日乘子。

【例 14.8】 利用函数 fminimax() 求解约束非线性规划问题。

目标函数:

```

function f = fminimaxfun(x)
f(1) = 2*x(1)^2 + x(2)^2 - 8*x(1) - 4*x(2);
f(2) = -x(1)^2 - x(2)^2 + 6;
f(3) = x(1) + 3*x(2) - 8;
f(4) = -x(1) + x(2) + 2;
f(5) = 5*x(1) + x(2) + 2;

```

利用函数 fminimax() 求解约束非线性规划问题。

```

>> x0 = [0.1; 0.1];
[x, fval, maxfval, exitflag, output, lambda] = fminimax(@fminimaxfun, x0) % 约束非线性规划问题的求解
Optimization terminated: magnitude of search direction less than 2*options.TolX
and maximum constraint violation is less than options.TolCon.
Active inequalities (to within options.TolCon = 1e-006):
    lower      upper      ineqlin      ineqnonlin
           2
           4
           5

x =
-0.000000000000000
1.56155281280883
fval =
Columns 1 through 3
-3.80776406404415  3.56155281280883 -3.31534156157351
Columns 4 through 5

```



```
3.56155281280883    3.56155281280883
maxfval =
    3.56155281280883
exitflag =
    4
output =
    iterations: 7
    funcCount: 39
    stepsize: 1
    algorithm: 'minimax SQP, Quasi-Newton, line_search'
firstorderopt: []
cgiterations: []
    message: [1x142 char]
lambda =
    lower: [2x1 double]
    upper: [2x1 double]
    eqlin: [0x1 double]
    eqnonlin: [0x1 double]
    ineqlin: [0x1 double]
    ineqnonlin: [0x1 double]
```

14.5 本章小结

本章主要介绍了 MATLAB 优化工具箱的使用，包括线性规划、整数规划、无约束规划和约束规划等常规的优化算法。通过本章的学习读者将掌握使用 MATLAB 优化工具箱求解目标函数的最小值、求解等规划问题。



第15章

曲线拟合工具箱

本章将介绍 MATLAB 曲线拟合工具箱的使用。曲线拟合应用非常广泛，在人们对某一未知领域内的规律探索中，需要建立相应的数学模型，而模型中往往含有一些需要估计的参数，要确定这些参数，就要用到曲线拟合。

15.1 曲线拟合工具箱简介

在数据处理中，求取自变量 x 和因变量 y 之间的近似函数关系表达式 $y=f(x)$ 的问题就是曲线拟合。MATLAB 曲线拟合工具箱 Curve Fitting Toolbox 提供了两种方法进行曲线拟合：一种是编程形式，需要编写代码利用命令对数据进行拟合，另一种是利用图形界面窗口进行操作。

MATLAB 曲线拟合工具箱功能非常强大，包括以下几项。

- 用于曲线和曲面拟合的图形工具。
- 使用自定义方程求解线性和非线性回归。
- 具有优化起始点和解算参数的回归模型库。
- 插值方法，包括 B 样条、薄板样条和张量积样条。
- 平滑方法，包括平滑样条、局部回归、Savitzky-Golay 滤波和移动平均。
- 预处理过程，包括偏值移除、分段、缩放和加权数据。
- 后处理过程，包括插值、外推、置信区间、积分和导数。

下面详细介绍曲线拟合工具箱常用功能。

15.2 利用图形界面进行曲线拟合

MATLAB 曲线拟合工具箱提供了可视化的图形界面进行曲线拟合，图形界面操作简单、方便，无须编写复杂的代码，对于希望快速利用 MATLAB 进行曲线拟合的入门读者尤为有用。

本节以数据拟合的一般流程为主线，结合实例详细介绍了如何利用图形界面进行曲线拟合。通过本节的介绍，将使读者对利用 MATLAB 进行曲线拟合的过程有感性认识，初步掌握这一工具图形界面的使用。

15.2.1 打开曲线拟合工具箱

在 MATLAB 的 Command 窗口中，输入 `cftool` 命令打开曲线拟合工具箱，如图 15.1 所示。其中主要有 5 个按钮，具体功能介绍如下。在后续章节中将详细介绍各按钮功能的使用。

- Data 按钮：导入待拟合的数据及对原始数据进行平滑预处理。

- Fitting 按钮：曲线拟合，计算拟合误差等统计参数。
- Exclude 按钮：在拟合数据中去除异常点。
- Plotting 按钮：显示拟合曲线。
- Analysis 按钮：做内插、外推、微分及积分拟合。

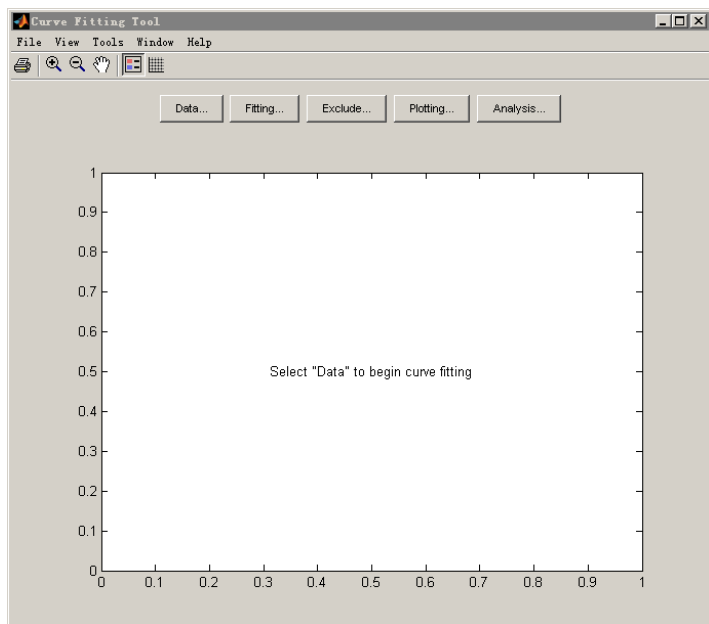


图 15.1 曲线拟合工具箱窗口

15.2.2 导入拟合数据集

导入数据前应注意数据必须已存放于 MATLAB 的 workspace 工作空间内，且待导入的自变量与因变量必须具有相同的维数，无穷大和不定值在拟合过程中将被忽略。数据的导入在前面基础知识的章节中已详细的介绍过，在此不再展开。

下面以 MATLAB 自带的文件 census.mat 为例，来演示曲线拟合的过程。census 数据包含两个变量：cdate 是列向量，对应 1790—1990 年，间隔 10 年；pop 是相应年份的美国人口数量。

(1) 单击曲线拟合工具箱窗口中的“Data”按钮，打开“Data”窗口。选择“Data Sets”选项卡，其中“Import workspace vectors”区域用于数据导入：“X data”下拉列表框用于选择曲线拟合自变量数据；“Y data”下拉列表框用于因变量数据的选择；“Weight”下拉列表框用于设置权重因子，如果不导入权重因子则默认全部数据权重为 1。这里在“X Data”下拉列表框中选择变量名“cdate”；“Y Data”下拉列表框中选择变量名 pop；设置数据集名为 pop vs.cdate，在“Preview”区域中将显示数据的预览图，如图 15.2 所示。

(2) 单击“Create data set”按钮，可设置拟合数据名称；Data sets 以列表的形式显示拟合数据集。当选定一个数据集后，可以对其进行查看（View 按钮）、重命名（Rename 按钮）、删除操作（Delete 按钮）；Preview 区域对所选数据进行图形预览。导入数据后在 MATLAB 的 workspace 窗口中，将出现变量 cdate 和 pop。

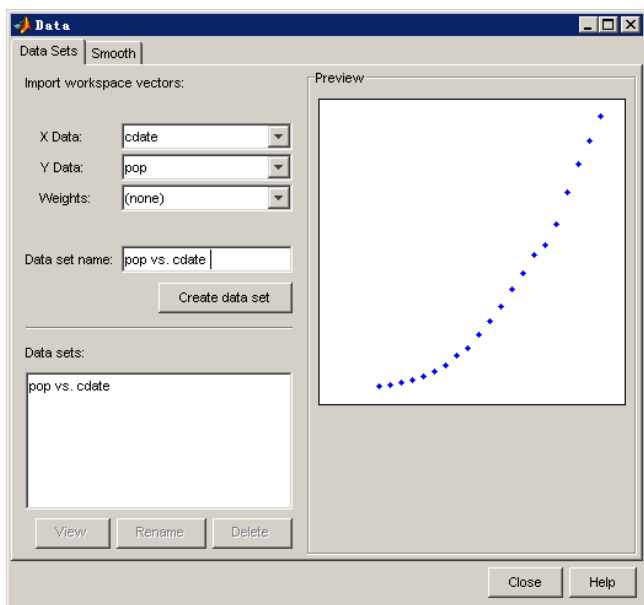


图 15.2 “Data”窗口中的“Data Sets”选项卡

15.2.3 拟合数据预处理

如需对原始数据进行平滑处理，可以选择“Data”窗口中的“Smooth”选项卡，如图 15.3 所示。

(1) 在“Original data set”下拉列表框中选择“pop vs.cdate”，设置需要拟合的数据集。默认在“Smoothed data set”区域中将显示平滑后的数据集名称 pop vs.cdate (smooth)。

Method 下拉列表框用于选择平滑数据的方法，包括以下几项。

- Moving Average: 移动平均滤波。
- Lowess: 局部加权散点图平滑数据，通过线性最小二乘和一阶多项式拟合。
- Loess: 局部加权散点图平滑数据，通过线性最小二乘和二阶多项式拟合。
- Savitzky-Golay filtering: 使用未加权线性最小二乘拟合指定阶数多项式过滤。如果选择了 Savitzky-Golay filtering，需要同时在出现的 Degree 下拉列表框中选择多项式阶数。
- Robust Lowess: 耐数据异常点的 Lowess 平滑。
- Robust Loess: 耐数据异常点的 Loess 平滑。

(2) 此处选择“Moving Average”平滑方法。在“Span”文本框中设置平滑计算的数据点的数目为 5，即每隔 5 个数据点进行平均。

(3) 单击“Create smoothed data set”按钮，在“Smoothed data sets”区域将出现产生的平滑数据集名 pop vs.cdate(smooth)。单击“View”按钮打开“View Data Set”窗口，可观察平滑后的数据，如图 15.4 所示。在“View Data Set”窗口左边将以图形化方式显示平滑后数据，右边以数据列表方式显示平滑后的数据数值。

(4) 返回“Smooth”选项卡。单击“Rename”按钮，可对平滑后数据集重新命名；单击“Delete”按钮，可删除平滑后的数据集；单击“Save to workspace”按钮，可保存平滑后的数据集到 MATLAB 的 Workspace 空间中。

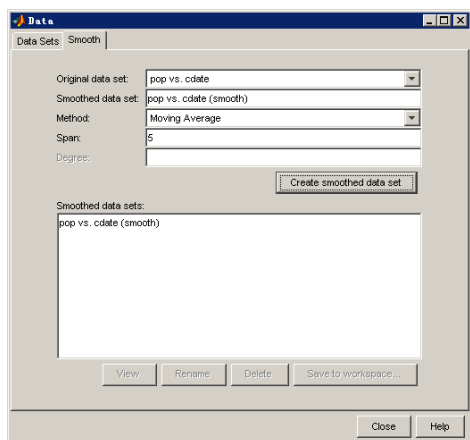


图 15.3 “Data”窗口中的“Smooth”选项卡

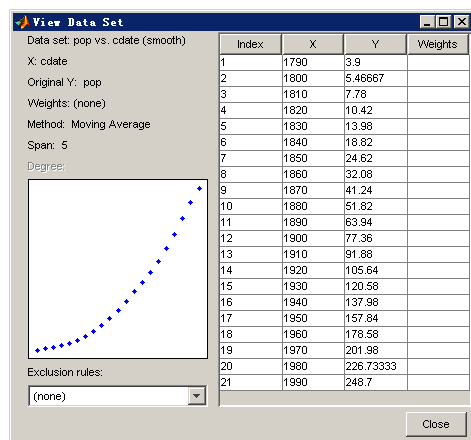


图 15.4 “View Data Set”窗口

15.2.4 曲线拟合

本节继续以 census 数据演示曲线拟合的过程。

(1) 返回曲线拟合工具箱窗口。单击“Fitting”按钮，弹出“Fitting”窗口，如图 15.5 所示。

“Fitting”窗口中包含“Fit Editor”和“Table of Fits”两个区域。“Fit Editor”区域用于定义拟合的文件名、拟合数据集、排除异常数据的规则、探索比较各种拟合方法对当前数据集的拟合效果，以及更改默认的拟合参数等；“Table of Fits”区域用于列出所有的拟合结果，保存或删除拟合结果。

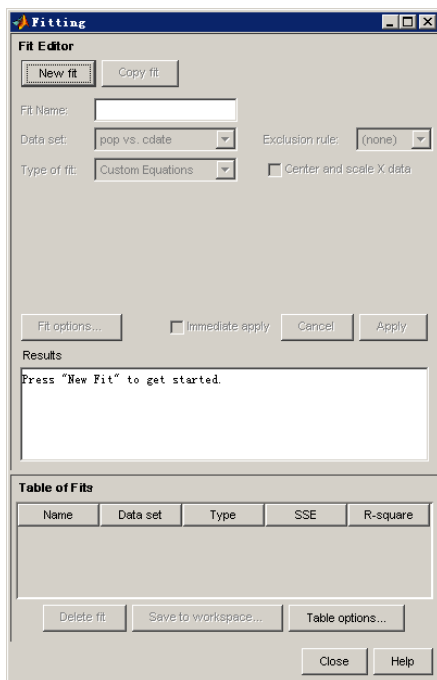


图 15.5 “Fitting”窗口

(2) 单击“New fit”按钮开始曲线拟合，如图 15.6 所示。其中，“Fit Name”选项默认为当前拟合曲线名称 fit 1；“Data set”下拉列表框中为当前的数据集 pop vs.cdate；“Exclusion rule”下拉列表框为排除异常值的规则 none。选择“Center and scale X data”复选框对数据进行中心化和离散化处理；“Type of fit”下拉列表框用于选择拟合的类型，包括参数和非参数拟合，具体

有 Custom Equations(自定义拟合模型)、Exponential(指数模型)、Fourier(傅里叶模型)、Gaussian(高斯模型)、Interpolant(内插法)、Polynomial(多项式模型) 等。这里选择 Polynomial 模型的 linear polynomial 函数用于曲线拟合。各模型的基本形式见表 15.1, 具体函数详细描述如下。

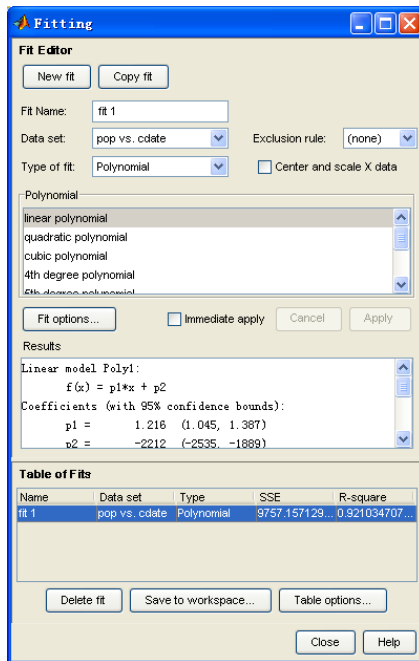


图 15.6 利用 “Fitting” 窗口进行曲线拟合

表 15.1 拟合模型类型

函数模型	基础形式
Custom Equations	用户自定义的函数类型
Exponential	$y = ae^{bx}$, $y = ae^{bx} + ce^{dx}$
Fourier	$y = a_0 + \sum_{i=1}^n a_i \cos(nwx) + b_i \sin(nwx)$
Gaussian	$y = \sum_{i=1}^n a_i e^{[-(\frac{x-b_i}{c_i})^2]}$
Polynomial	$y = \sum_{i=1}^{n+1} p_i x^{n+1-i}$
Power	$y = ax^b$, $y = a + bx^c$
Rational	$y = \frac{\sum_{i=1}^{n+1} p_i x^{n+1-i}}{x^m + \sum_{i=1}^m q_i x^{m-i}}$
Sum of Sin Functions	$y = \sum_{i=1}^n a_i \sin(b_i x + c_i)$
Weibull	$y = abx^{b-1} e^{-ax^b}$

- Custom Equations: 自定义函数形式, 包括线性和非线性模型。
- Exponential: 指数模型拟合, 分为单指数和双指数模型。
- Fourier: 傅里叶模型拟合, 使用正弦函数与余弦函数之和拟合, 共 8 个多项式。
- Gaussian: 高斯模型拟合, 有 8 种类型。
- Polynomial: 多项式模型, 包括一阶到九阶多项式。
- Power: 幂函数模型, 包括 $y = ax^b$ 和 $y = a + bx^c$ 两种。
- Rational: 有理拟合, 两个多项式之比, 分子、分母共有的类型是 linear polynomial、quadratic polynomial、cubic polynomial、4–5th degree polynomial, 此外, 分子还包括 constant 型。
- Sum of Sin Functions: 正弦曲线, 在 $y = \sum_{i=1}^n a_i \sin(b_i x + c_i)$ 公式中, a 为振幅, b 为频率, c 为常数项, n 小于等于 8, 共有 8 个模型可选。
- Weibull: 双参数威布尔分布, 是一种单峰的正偏态分布函数, 公式 $y = abx^{b-1}e^{-ax^b}$ 中, a 为尺度因子, b 为形状因子。
- Interpolant: 内插法, 可用于非参数拟合, 包括 linear (线性插值)、nearest neighbor (最近插值)、cubic spline (三次样条插值) 和 shape-preserving (分段三次艾尔米特内插)。

(3) 单击“Fit options”按钮, 在弹出的“Fit options for poly1”对话框中进行拟合参数的设置, 可设置拟合算法、修改待估计参数的上下限等参数, 如图 15.7 所示。其中, Robust 选项用于设置拟合是否采用稳健的参数估计方法, 这里选择“Off”选项, 表示未用稳健回归, 同时可以在对话框中设置拟合参数 p1 和 p2 的范围。设置完成后单击“Close”按钮, 返回到“Fitting”窗口。

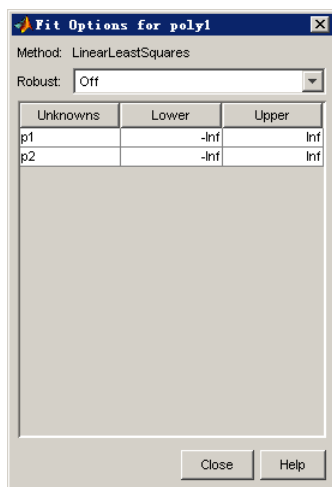


图 15.7 拟合参数设置

(4) 在“Fitting”窗口中单击“Apply”按钮, 采用选定的方法对数据进行曲线拟合, 在 Results 区域将显示拟合的结果, 如图 15.6 所示。

曲线拟合的结果包括拟合函数形式, 各参数的置信区间, 反映拟合效果的统计参数。统计指标主要有 SSE(误差平方和)、R-square(决定系数)、Adjusted R-square(校正决定系数)、RMSE(均方根误差平方和), 其中 SSE 和 RMSE 越小, R-square 和 Adjusted R-square 越接近 1, 曲线拟合效果就越好。

本例中的曲线拟合结果如下。

```
Linear model Poly1:
    f(x) = p1*x + p2
Coefficients (with 95% confidence bounds):
```

```

p1 =      1.216  (1.045, 1.387)
p2 =     -2212  (-2535, -1889)
Goodness of fit:
SSE: 9757
R-square: 0.921
Adjusted R-square: 0.9169
RMSE: 22.66

```

线性模型系数 p_1 、 p_2 分别为 1.032 和 1.44，模型系数 95% 的置信区间为 $p_1(0.7774, 1.286)$ 和 $p_2(-0.06561, 2.946)$ ，拟合的 SSE、R-square、Adjusted R-square、RMSE 分别为 12.53、0.9034、0.8926 和 1.18。

(5) 在图 15.6 的“Table of Fits”区域可以对拟合结果进行操作，单击“Table options”按钮，弹出“Table options”对话框，设置拟合结果的显示，如图 15.8 所示。选择“Name”、“Data set”、“Type”、“SSE”、“R-square”复选框，单击“Close”按钮关闭“Table options”窗口，返回“Fitting”窗口。在“Table of Fits”区域将显示选择的拟合结果项。单击“Delete fit”按钮可删除拟合结果。单击“Save to workspace”按钮可以保存拟合的结果。

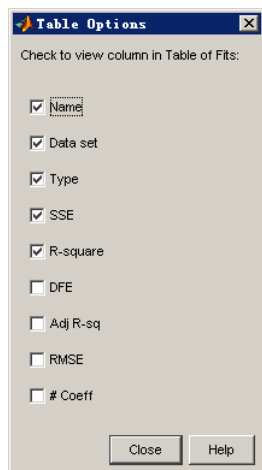


图 15.8 拟合结果设置

(6) 如果对于拟合的结果不满意，可以单击“New fit”按钮，选择不同的模型进行拟合，本例中分别选择了线性、二次、三次多项式进行曲线拟合。

最终曲线拟合结果的确定，除了可以比较“Fitting”窗口“Results”区域的统计指标结果外，在曲线拟合工具箱界面可以通过拟合的数据图形，更为直观地比较各模型拟合结果。选择“View”→“Prediction Bounds”命令，显示函数预测值的置信区间，同时选择“View”→“Confident Level”命令可设置置信区间显著性水平，如图 15.9 所示。残差图的显示通过选择“View”→“Residuals”命令设置，如图 15.10 所示。“None”、“Scatter Plot”、“Line Plot”三项分别表示不显示残差图、散点残差图、线性残差图。例子中选择了 Scatter Plot 残差图，在图 15.10 中出现了不同拟合形式的散点型残差图。

根据拟合的效果确定最佳的拟合模型，如果为实际工程中模型，同时需要考虑模型参数形式是否具有实际意义。本例中选择曲线拟合形式中，二次、三次多项式拟合效果优于线性拟合效果。

当拟合效果较差的时候，MATLAB 会给出警告：“Warnings during fitting:Equation is badly conditioned. Remove repeated data points or try centering and scaling.”，这时可以尝试去除重复数据点或中心化数据改善拟合效果。

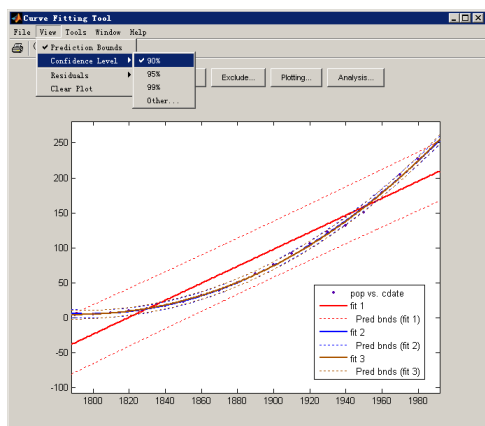


图 15.9 拟合结果图显示

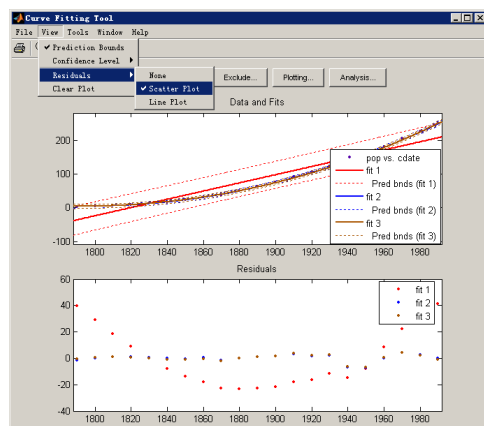


图 15.10 拟合结果残差图显示

15.2.5 异常数据的去除

用于曲线拟合的数据多为实际观测的数据，而在实际的观测中由于各种原因极易引入异常的数据点，拟合数据中的异常点会对最后的拟合结果造成很大影响。曲线拟合工具箱提供了较为方便地查找异常点的方法：排除法是对数据中的异常值进行排除；区间排除法是采用一定的区间去排除那些用于系统误差导致偏离正常值的异常值，下面介绍具体用法。

(1) 单击曲线拟合界面中的“Exclude”按钮，打开“Exclude”窗口，如图 15.11 所示。可通过定义异常点的准则来排除异常区间，例如本例演示数据中 Y 为美国的人口数，可以定义小于 0 的值为错误的异常点。同时“Exclude”窗口中还提供了通过图像直接排查异常点的功能。

(2) 单击“Exclude graphically”按钮，打开“Select Points for Exclusion Rule”窗口，如图 15.12 所示。其中蓝色圈的数据点表示已包含的数据。单击某一异常点，数据变为红色的叉，此时数据不参与曲线拟合过程。

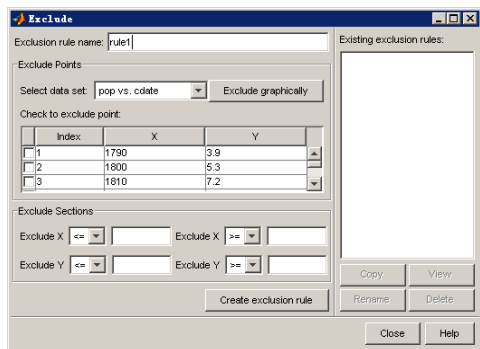


图 15.11 “Exclude”窗口

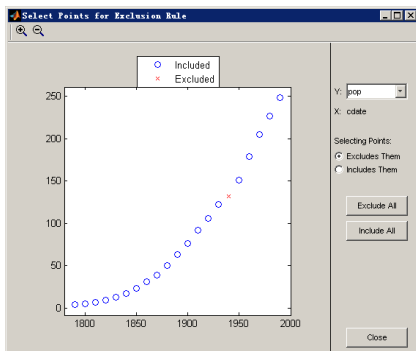


图 15.12 “Select Points for Exclusion Rule”窗口

15.2.6 绘图显示设置

当用多种函数形式对同一数据拟合后，除了可以从统计指标上判断各拟合结果的好坏外，还可以通过多种模型拟合结果图像的同时显示，以更为直观地显示拟合结果的差异。而拟合图像显示的控制，通过“Plotting”窗口来完成。

单击曲线拟合界面中的 Plotting 按钮，打开“Plotting”窗口，如图 15.13 所示。其中“Plot data sets”区域用于选择需要显示图像的拟合数据名，“Plot fits”区域用于选择需要同时显示的拟合图像。曲线拟合界面中的图像会随选择的显示图像的变化而同步变化。

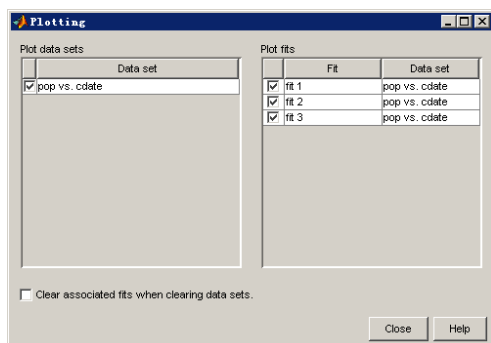


图 15.13 “Plotting” 窗口

15.3 使用命令行拟合数据

通过前面章节的学习，读者学习了如何通过曲线拟合工具箱进行拟合数据，在一般情况下利用工具箱拟合数据是比较方便的。但是有的时候需要拟合的曲线可能比较多，或者需要重复拟合曲线，此时可以利用命令行的方式来拟合数据。本节将主要介绍使用命令行拟合数据，下面具体介绍这些用于曲线拟合的函数。

1. fitytype()函数

fitytype()函数用于定义需要拟合的模型，其调用格式如下。

- `fitytype = fitytype('ltype')`: 使用曲线拟合工具箱内部已有的模型拟合数据，参数 `ltype` 为拟合的模型名称，可以从曲线拟合工具箱帮助文档中获取相关已有模型的信息。
- `fitytype = fitytype('expr')`: 用于拟合自定义模型，其中参数 `expr` 为用户自定义的模型的表达式，返回拟合模型 `fitytype`。
- `fitytype = fitytype('expr','PropertyName',PropertyValue,...)`: 定义曲线拟合模型，并设置模型相关属性，其中包括 `coefficients`（用于定义拟合模型返回系数的变量名称）、`dependent`（定义自变量名称）、`independent`（定义因变量名称）、`options`（当前模型的拟合参数设置）和 `problem`（定义模型中的常数变量）。

2. fitoptions()函数

fitoptions()函数用于设置模型拟合的参数，其调用格式如下。

- `opts = fitoptions`: 返回默认状态下的曲线拟合的参数设置，这些属性在各种曲线拟合中都存在，它们是 `Normalize`（数据是否归一化）、`Exclude`（异常点的去除）、`Weights`（拟合变量的权重）和 `Method`（拟合的方法）。
- `opts = fitoptions('ltype')`: 设置拟合的模型为 `ltype`。
- `opts = fitoptions('ltype','PropertyName',PropertyValue,...)`: 设置拟合模型 `ltype` 的相关参数，根据不同的模型将设置不同的参数。
- `opts = fitoptions('method',value)`: 设置拟合方法 `method` 及相应方法的值，其中方法 `NearestInterpolant`、`LinearInterpolant`、`PchipInterpolant` 和 `CubicSplineInterpolant` 没有额外的拟合参数需要设置。
- `opts = fitoptions('method',value,'PropertyName',PropertyValue,...)`: 设置曲线拟合的方法 `method` 及其参数值 `value`，同时可设置相应的属性。

- `opts = fitoptions(opts,'PropertyName',PropertyValue,...)`: 对原曲线拟合的参数更新新的参数值。
- `opts = fitoptions(opts,newopts)`: 以新的参数设置 `newopts` 更新老的参数设置。

3. fit()函数

`fit()`函数用于根据已知数据点进行曲线拟合，其调用格式如下。

- `fresult = fit(xdata,ydata,'ltype')`: 拟合自变量为 `xdata` 和因变量为 `ydata` 的 `ltype` 模型的系数，返回的参数 `fresult` 包括模型的表达式及其系数。
- `fresult = fit(xdata,ydata,'ltype','PropertyName',PropertyValue,...)`: 拟合模型的系数，并设置拟合模型的相关参数。
- `fresult = fit(xdata,ydata,'ltype',opts)`: 使用通过函数 `fitoptions()` 设置的模型参数拟合模型。
- `fresult = fit(xdata,ydata,'ltype',..., 'problem',values)`: 设置模型中常数变量 `problem` 的值 `values`。
- `fresult = fit(xdata,ydata,ftype,...)`: 使用用户指定的函数类型 `ftype` 拟合模型。
- `[fresult,gof] = fit(...)`: 拟合曲线，返回的参数包括拟合模型的系数 `fresult` 和模型误差估计的相关参数 `gof`，其中包括 `sse` (误差平方和)、`rsquare` (模型决定系数)、`dfe` (自由度)、`adjrsquare` (复相关系数) 和 `rmse` (平方根误差)。

4. coeffvalues()函数

`coeffvalues()`函数用于根据曲线拟合的模型获取模型系数，其调用格式如下。

- `coeffvalues(cfobj)`: 其中参数 `cfobj` 为利用函数 `fit()` 拟合的模型对象，函数返回模型的系数。
- 利用上述的 4 个函数，用户可以完成基本的曲线拟合的过程，其拟合模型的基本过程如下：

(1) 利用函数 `fitttype()` 定义拟合模型的类型，可以选择曲线拟合工具箱自带的函数或者为用户自定义的模型。

(2) 利用函数 `fitoptions()` 设置曲线拟合的相关参数。

(3) 利用函数 `fit()` 根据函数 `fitttype()` 定义拟合模型类型和函数 `fitoptions()` 设置曲线拟合的相关参数，进行曲线拟合。

(4) 利用函数 `coeffvalues()` 进一步获取模型的系数。

下面以一个具体的实例演示利用这些函数进行曲线拟合的过程。

【例 15.1】 利用曲线拟合工具箱的函数进行曲线拟合。

```
%建立曲线拟合的数据
load census
plot(cdate,pop,'b+'),hold on
% 利用曲线拟合工具箱函数进行拟合
ftype=fitttype('poly2')
fopt=fitoptions('Method','linearLeastSquares')
[fresult,gof]=fit(cdate,pop,ftype,fopt)
covalue = coeffvalues(fresult)
%根据拟合的参数绘图
yp=covalue(1).*cdate.^2+covalue(2).*cdate+covalue(3);
plot(cdate,yp,'ro')
legend('精确值','预测值')
hold off
```

执行上述代码，在命令窗口返回的结果包括以下内容。

(1) 拟合的模型类型如下。

```
ftype =
Linear model Poly2:
ftype(p1,p2,p3,x) = p1*x^2 + p2*x + p3
```

(2) 拟合模型的参数设置情况如下。

```
fopt =
    Normalize: 'off'
    Exclude: []
    Weights: []
    Method: 'LinearLeastSquares'
    Robust: 'Off'
    Lower: [1x0 double]
    Upper: [1x0 double]
```

(3) 拟合模型的结果，其中包括拟合的模型的系数及其置信区间。

```
fresult =
    Linear model Poly2:
    fresult(x) = p1*x^2 + p2*x + p3
    Coefficients (with 95% confidence bounds):
    p1 =    0.006541 (0.006124, 0.006958)
    p2 =   -23.51 (-25.09, -21.93)
    p3 =  2.113e+004 (1.964e+004, 2.262e+004)
```

(4) 拟合模型的参数估计如下。

```
gof =
    sse: 1.590292991767960e+002
    rsquare: 0.99871296577201
    dfe: 18
    adjrsquare: 0.99856996196890
    rmse: 2.97236624011537
```

(5) 返回模型拟合系数如下。

```
covalue =
    1.0e+004 *
    0.00000065411305 -0.00235097459954  2.11295921192324
```

执行上述程序，将生成如图 15.14 所示的曲线拟合效果图，其中“+”形的点为模型实际值，“o”形的点为模型预测值。

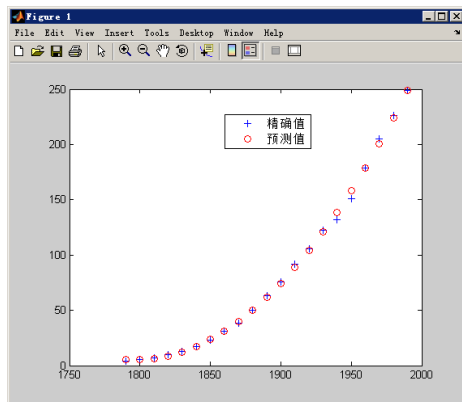


图 15.14 曲线拟合效果图

15.4 本章小结

本章主要向读者介绍了 MATLAB 曲线拟合工具箱的使用，主要包括曲线拟合工具箱简介、利用 GUI 界面进行曲线拟合和利用命令行函数法进行曲线拟合。其中利用曲线拟合工具箱的界面操作进行曲线更为方便，但是对于需要重复操作的情况，利用命令行的方式更为普遍。通过本章的学习读者将基本掌握利用 MATLAB 进行曲线拟合的基本操作。



第16章

神经网络工具箱

本章主要介绍 MATLAB 神经网络工具箱的使用。近年来，随着神经网络算法的应用领域的扩展，基于 MATLAB 求解神经网络问题，由于操作简单，有直接的函数调用或者可以直接通过界面操作完成，同时还可以根据用户的需要设计不同的网络，使原本复杂的神经网络算法可以简单实现，这些都促进了神经网络算法的应用。

人工神经网络包括 BP 神经网络、径向基神经网络、自组织神经网络、广义回归神经网络等，本章将依次介绍这些网络的函数命令实现方法。其中，以 BP 神经网络最为常用，本章将重点介绍该神经网络的使用。最后一节将介绍如何利用工具箱的 GUI 界面实现神经网络。

通过本章的学习读者将对神经网络的算法有基本了解，了解神经网络能干什么，明确什么问题可以使用神经网络求解，并尝试使用各神经网络解决问题。

16.1 人工神经网络介绍

人工神经网络 (Artificial Neural Network, ANN) 通过模拟生物神经元的工作原理，来训练网络的学习算法，通过大量的处理单元互连的数学模型来实现。

16.1.1 人工神经网络的基本特征

人工神经网络的基本特征有以下 5 个方面。

1. 处理的非线性

人工神经网络通过神经元的激活或抑制的不同状态，可求解数学上的非线性问题。可以处理非线性问题是人工神经网络模型很重要的基本特征。

2. 处理的并行性

神经网络模型依赖于各个神经元的协同工作，网络中同一层的神经元可以同时工作，说明了神经网络具有较好的并行性。处理的并行性对许多复杂问题的求解是很有利的，可以提高解决问题的效率。

3. 信息处理与存储并存

神经网络中每个神经元同时具有信息处理和信息存储的功能。神经元在对信息计算处理的同时，可以有效地记忆信息，在下一步的处理中可以使用记忆的信息调节处理。

4. 处理的学习性

权值在神经网络模型中用于表述神经元的连接强度，通过模型对样本的学习，将调整这个权值，从而能根据样本的特征调整神经元之间的连接强度，使神经元对样本有较好的学习性。当神经网络模型通过学习掌握了某样本的特征后，当用新的数据输入模型，也能获得较为可靠的结果。

5. 一定的鲁棒性和容错性

由于神经网络的结果是多个神经元共同作用的,因而网络模型具有较好的鲁棒性和容错性。个别输入样本的错误信息对最后的结果影响不大。

16.1.2 人工神经网络的分类

目前已有几十种不同的神经网络模型,按照不同的角度对神经网络进行分类,可以分以下 3 种。

1. 按照网络的结构分

- 前向网络。
- 反馈网络。

2. 按照学习方式分

- 有教师学习网络。
- 无教师学习网络。

3. 按照网络性能方式分

- 连续型和离散型网络。
- 随机型和确定型网络。

其中,以 BP 神经网络、径向基神经网络、自组织神经网络、广义回归神经网络等在工程等应用领域最为常用,下面详细介绍这几种网络模型。

16.1.3 人工神经网络的应用

目前人工神经网络的应用已在各个领域得到了广泛的应用,主要体现在以下几个方面。

1. 自动控制领域

控制领域常会遇到的问题是非线性、不确定和极其复杂的系统,而且控制领域的求解又往往需要算法能适应这种不确定性,并能随环境的变化而变化。传统的建模技术不能满足这一要求,因而现在控制系统的很多问题越来越依赖于现代的人工智能的算法(如人工神经网络)来求解。

2. 模式识别

近年来,模式识别是一个相当热的研究领域,其中算法研究更是模式识别研究的重点之一。人工神经网络由于具有较好的学习能力、一定的容错性等特点,被证明具有较强的分类识别能力,已被广泛用于模式识别领域,并取得了很好的分类精度。

3. 图像处理

人工神经网络可以很好地模拟大脑的功能,这就决定了它可以让计算机像大脑一样分析、处理图像。运行神经网络对图像分类,检测图像的边缘,目前已应用得很好,而且相信随着神经网络算法的不断完善,也会在图像处理领域发挥更大的作用。

4. 故障诊断

由于人工神经网络具有较好的非线性映射能力,在故障诊断中具有较好的应用。故障数据可以作为神经网络的输入,而输出可以为故障的原因,从而实现故障诊断。

5. 预测预报

神经网络具有一定的泛化能力,其预测预报能力可被用于气象灾害、病虫害预报、金融投资、股票、电力系统预测等各种需要预测预报的领域。

综上,我们可以看到近年来随着人工神经网络技术的不断成熟,其应用领域也不断扩大,一

般的需要分类、拟合、预测的领域都可以应用人工神经网络。上面所列举的仅是目前一些神经网络比较成功的应用领域，读者可以结合自身的应用领域，考虑人工神经网络是否有其应用的实际价值。

16.1.4 MATLAB 人工神经网络工具箱

MATLAB 神经网络工具箱以神经网络的基本理论为基础，提供了常用神经网络求解的函数，函数可以设置不同的激活函数，以及权值、阈值等网络参数。对于用户来说无须再编写复杂的算法，可以集中精力训练网络，考虑如何使网络的精度、稳定性更好。相对而言，利用 MATLAB 神经网络工具箱求解神经网络问题与其他编程语言相比，是最简单、效率最高的。

16.2 BP 神经网络

反向传播算法（Backpropagation Algorithm，BP）是目前最为常用的神经网络模型。该网络模型较为简单，且已被证明只要网络规模足够大，即网络中隐含层节点数足够多，网络能够以任意精度逼近任意的连续函数。本节将具体介绍 BP 神经网络的基本原理、特点、应用及其实现。

16.2.1 BP 神经网络基本原理

BP 神经网络是一种多层前馈网络，具有信息前向传递，误差反向传递的特点。BP 神经网络模型如图 16.1 所示，模型由输入层、隐含层和输出层构成。信号由输入层传入网络模型，经隐含层的处理后，传至输出层，此过程为信息的前向传递。同时，输出层的输出与期望输出之间会产生误差，误差将反向传递，调节网络的权值与阈值，此过程即为误差的反向传递。

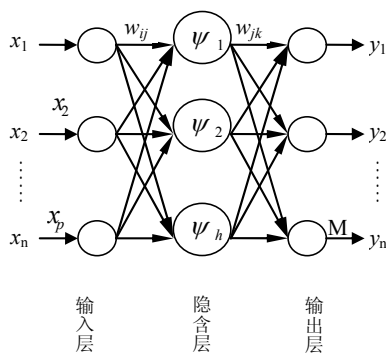


图 16.1 BP 神经网络模型

16.2.2 BP 神经网络的特点

根据 BP 神经网络的基本原理，可以发现 BP 神经网络具有以下主要特点。

1. 良好的学习能力

BP 神经网络通过对误差的反向传递，可以有效地调整模型的权值和阈值，使模型能学习到样本数据的特征。

2. 非线性映射能力

BP 神经网络能完成输入到输出的非线性映射，且这种映射无须有明确的数学表达式。网络非

线性映射能力需要通过大量样本数据的训练来获得。

3. 泛化能力

泛化能力是指当训练好的网络有未曾训练过的数据输入时，网络模型也能准确的映射输出。BP 神经网络模型具有一定的泛化能力。

4. 容错能力

BP 神经网络模型具有一定的容错能力，当输入样本中有个别误差较大的数据对网络模型整体的影响很小。

16.2.3 BP 神经网络的应用

由于 BP 神经网络对样本数据良好的学习能力、非线性映射能力、一定的泛化能力和容错能力，使得 BP 神经网络成为神经网络中最为常用的网络模型。BP 神经网络在各领域的应用可以总结为以下 3 方面。

1. 分类识别

通过学习样本数据的类型特征，进行分类。

2. 回归预测

通过学习已知样本数据输入/输出特征对未知样本输入预测其输出。

3. 函数逼近或插值

通过对输入和期望的输出规律的学习，对函数逼近或对未知点进行插值。

16.2.4 BP 神经网络的实现

本小节将向读者介绍如何利用 MATLAB 工具箱实现 BP 神经网络。包括神经网络结构设计、神经网络参数设置、神经网络模型的构建与训练。

1. 神经网络结构的设计

实现 BP 神经网络需要设计合理的网络结构，包括输入层、隐含层、输出层及其各层之间的传递函数的设计。

1) 输入层

网络的输入层应是能最好反映样本数据变化特征的数据，神经网络精度的高低很大程度上取决于选择的样本是否具有较好的代表性、是否有足够多的高精度的样本作为神经网络的输入。对于输入层变量个数过多的情况，可以使用主成分分析等数据压缩方法，提取关键信息。输入层节点数即为输入数据的变量个数。一般要求输入数据进行归一化，把数据归一化到 $-1 \sim 1$ 之间，取消各变量数量级的差别。在 MATLAB 7.0 中可用于数据归一化的函数为 `premnmx()`，其调用格式如下。

- `[PN,minp,maxp] = premnmx(P)`: 对输入数据 P 进行归一化操作，矩阵 P 的行为各变量，列为各样本，即矩阵 P 的大小为输入节点数 \times 样本数，函数返回归一化后的数据 PN，各变量的最小值 minp 和最大值 maxp。
- `[PN,minp,maxp,TN,mint,maxt] = premnmx(P,T)`: 同时对输入数据和输出数据进行归一化，其中矩阵 P 和 T 要求具有相同的列数，即样本数，矩阵 P 和 T 的行数分别为输入节点数和输出节点数，函数返回归一化后的数据。

2) 隐含层

在设计 BP 神经网络时需要设置隐含层的层数和节点数，其中隐含层的层数属于一般的问题，

隐含层为 1 的 BP 神经网络已被理论证明在不限神经网络节点数的同时, 可以实现任意非线性映射, 所以一般情况下, 三层的 BP 神经网络 (输入层+隐含层+输出层) 即可达到用户的需要。

关于隐含层节点数的确定目前也没有通过理论来确定, 需要用户通过试错法来确定最为适宜的隐层节点数。隐含层节点数的选取直接关系到神经网络模型设计的好坏, 一方面, 增加隐含层节点数可以提高模型精度。另一方面, 隐含层节点数过多会大量增加计算量, 导致学习实践过长, 也易使模型陷入过拟合。因而在一般情况下, 在能解决问题的时候, 不推荐增加过多的隐含层节点数。

隐含层节点数对于神经网络模型的好坏很关键, 对于入门的用户可能在初次设计神经网络模型的时候没有经验, 这里给出前人经验总结的隐层节点数确定的公式:

$$n = \sqrt{m_1 + m_2} + a$$

其中, n 为隐含层节点数, m_1 和 m_2 分别为输入层和输出层的节点数, a 为不确定参数, 是 1 ~ 10 之间的常数。

3) 输出层

输出层即为神经网络的期望输出, 输出变量个数为输出层节点数。当对输入、输出数据归一化后, 网络训练得出的模型预测值也为归一化后的数据, 此时需要对数据反归一化。在 MATLAB 7.0 中数据反归一化的函数为 `postmnmx()`, 其调用格式如下。

- `[P,T] = postmnmx(PN,minp,maxp,TN,mint,maxt)`: 函数的输入参数为归一化后的数据和相应的归一化时的最大值和最小值, 函数返回未归一化的数据。
- `[p] = postmnmx(PN,minp,maxp)`: 实现矩阵 PN 数据的反归一化。

4) 传递函数

传递函数用于连接输入层和隐含层、隐含层和输出层。BP 神经网络提供的传递函数有如下几种。

- 函数 `tansig()`: 把神经元的输入数据范围从 $(-\infty, +\infty)$ 映射到 $(-1, 1)$ 。
- 函数 `logsig()`: 把神经元的输入数据范围从 $(-\infty, +\infty)$ 映射到 $(0, 1)$ 。
- 函数 `purelin()`: 线性映射, 输出数据可以为任意值。

上述 3 种传递函数的示意图如图 16.2 所示。

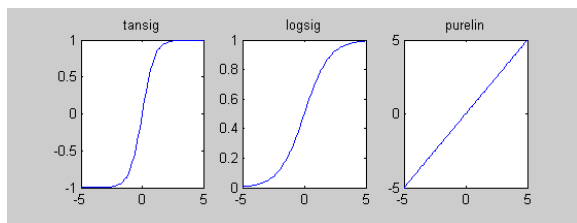


图 16.2 BP 神经网络传递函数

5) 训练方法

BP 神经网络模型提供了多种方法训练函数, 其中常用的有如下 3 种。

- `traingd` 算法 (标准的梯度下降算法): 基本的 BP 训练算法。
- `traingdm` 算法 (有动量的梯度下降算法): 算法在权值调整过程中考虑梯度方向。
- `trainгда` 算法 (自适应学习步长法): 学习步长可以根据网络误差性能函数进行自动调节。
- `trainrp` 算法 (弹性 BP 算法): 该算法可消除偏导数的大小对权值的影响, 只根据导数的符号更新权值而不考虑导数的大小。
- 四种变梯度类算法: `traincgf` (Fletcher-Reeves 共轭梯度算法)、`traincgp` (Polak-Ribiers 共轭梯度算法)、`traincgb` (Powell-Beale 共轭梯度算法) 和 `trainscg` (成比例的共轭梯度

算法)与普通的梯度下降法相比具有较快的收敛速率。

- `trainbfg` 算法 (BFGC 拟牛顿算法): 该算法计算量和存储量大, 适合于小型网络。
- `trainlm` 算法 (Levenberg Marquardt): 该算法学习速度较快, 但占用内存大, 适用于中等规模网络, MATLAB 神经网络工具箱默认采用此训练算法。

6) 学习方法

可以设置的学习方法有如下两种。

- `learngd` 学习方法: 梯度下降权值和阈值学习规则。
- `learngdm` 学习方法: 带动量项的梯度下降权值和阈值学习规则。

7) 误差性能函数

可以设置的误差性能函数有如下两种。

- `mse`: 计算输出变量与期望变量间的均方误差。
- `msereg`: 规则化均方误差函数, 计算输出误差的同时考虑了权值和阈值的均方和。

2. 神经网络参数的设置

在 BP 神经网络网络训练时, 可以设置一定的训练参数, 使网络具有更好的表现。根据所采用的不同网络训练方法, 可以设置的参数也有差异, 其中函数 `traingd` 有关可以设置的神经网络参数有如下几种。

- `net.trainParam.epochs`: 网络的最大训练次数, 即在网络未达到设置的误差下也停止训练, 默认为 1000。
- `net.trainParam.goal`: 网络训练达到的精度, 默认为 0。
- `net.trainParam.lr`: 学习率, 默认为 0.01, 为提高学习速度, 应采用大的学习率。但学习率太大却可能导致网络不收敛。
- `net.trainParam.max_fail`: 最大失败次数, 默认为 6。
- `net.trainParam.min_grad`: 最小梯度要求, 默认为 $1e-10$ 。
- `net.trainParam.show`: 控制在训练的时候显示迭代过程, NaN 表示不显示。
- `net.trainParam.time`: 最大训练时间, 默认为 inf。

3. 神经网络模型的构建与训练

MATLAB 中用于构建神经网络的函数为 `newff()`, 其调用格式如下。

`net = newff(PR,[S1 S2...SN],{TF1 TF2...TFN1},BTF,BLF,PF)`: 其中, 矩阵 PR 为网络输入向量取值范围, 矩阵的大小为输入变量数 $\times 2$, 其中第一列为各输入变量的最小值, 第二列为各输入变量的最大值; [S1 S2...SN]依次表示网络各隐含层和各输出层的节点数; {TF1 TF2...TFN1}表示网络输出层和隐含层, 隐含层和输出层的传递函数, 默认为'tansig'; BTF 表示网络的训练函数, 默认为'trainlm'; BLF 表示网络的权值学习函数, 默认为'learngdm'; PF 表示网络性能评价函数, 默认为'mse'; 函数返回建立的网络 net。

MATLAB 中用于训练神经网络的函数为 `train()`, 其调用格式如下。

`[net,tr,Y,E,Pf,Af] = train(net,P,T,Pi,Ai)`: 其中, net 为构建好的神经网络; P 为输入参数, 其大小为输入变量数 \times 样本数; T 为目标输出矩阵, 其大小为输出变量数 \times 样本数; Pi 为初始化输入层条件, 默认为 0; Ai 为初始化输出层条件, 默认为 0; 返回训练好的网络 net; tr 为训练过程记录; Y 为网络的输出; E 为网络的误差; Pf 为最终的输入层条件; Af 为最终的输出层条件。

MATLAB 中用于预测神经网络的函数为 `sim()`, 其调用格式如下。

`Y= sim(net,P)`: 其中, net 为训练好的神经网络; P 为网络预测的输入参数, 要求与训练的网

络有相同的输入变量数；返回网络的预测输出 Y。

【例 16.1】BP 神经网络程序演示。

利用 BP 神经网络逼近正弦函数。

```
P=0:2*pi/20:2*pi;
T=sin(P);
net=newff(minmax(P),[5,1],{'logsig','purelin'})
%设置网络训练的参数
net.trainParam.epochs=10000;
net.trainParam.goal=0.00001;
net.trainParam.lr=0.01;
[net,tr] = train(net,P,T);
y=sim(net,P);
%查看网络逼近正弦函数的效果
figure;
plot(P,T);
hold on;
plot(P,y,'ro')
```

%构建正弦函数
%构建 BP 神经网络
%最大训练次数
%网络收敛目标
%网络学习率
%训练网络
%网络预测

运行上述程序将生成如图 16.3 所示的函数逼近的 BP 神经网络误差变化图，并做如图 16.4 所示的 BP 神经网络的函数逼近效果图，显示网络的结果。

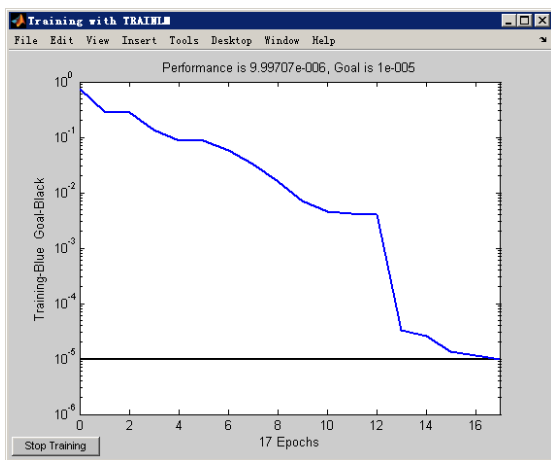


图 16.3 函数逼近的 BP 神经网络误差变化

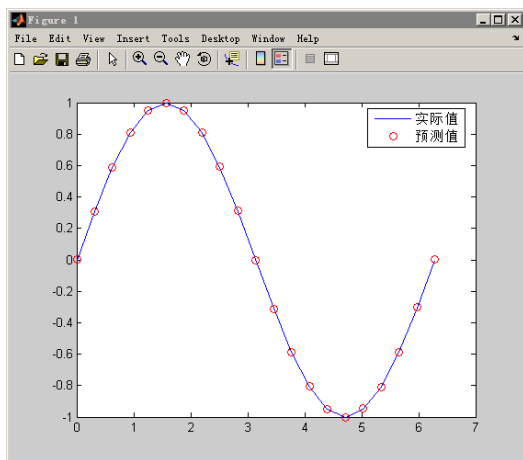


图 16.4 BP 神经网络的函数逼近效果

16.2.5 BP 神经网络的不足

BP 神经网络虽然得到了人们广泛的使用，但在应用的过程中也发现了以下一些不足。

1. 网络收敛速度慢

BP 神经网络的收敛速度较慢，这主要是因为为了保证 BP 神经网络的稳定性，算法一般采用比较小的学习率，导致网络收敛速度较慢。对于一些复杂的问题更是需要花费大量的时间训练网络。

2. 网络易陷入局部最小点

BP 神经网络的误差曲面可能存在多个局部的极小点，BP 神经网络易陷入局部最小点，网络可能在没有达到全局最小点的时间就停止训练。

3. 网络学习和记忆不稳定

当有新的样本需要训练好的网络学习的时候, 必须将原来的样本和新的样本放在一起重新训练, 之前训练好的网络将被破坏。

4. 隐含层层数和节点数无法准确确定

目前 BP 神经网络的隐含层层数和节点数无法准确确定, 而隐含层的结构又会对网络模型的准确性造成很大的影响。而根据经验法或者程序员试错确定的参数, 很可能并不是最佳的隐含层结构。

BP 神经网络自身存在以上的不足, 因而用户在使用 BP 神经网络求解问题的时候, 如果计算的结果不好或者不合理, 可以考虑是否是上述原因造成的。同时, 近年来对 BP 神经网络的改进和优化算法也层出不穷, 其中结合遗传算法、粒子群算法等的 BP 神经网络, 由于能优化获得较好的参数, 构建的网络模型更为稳定可靠, 感兴趣的读者可以阅读相关文献资料, 尝试建立优化的 BP 神经网络。

16.3 径向基神经网络

径向基神经网络是一种前向型的神经网络, 由输入层、隐含层和输出层组成。其隐含层为径向基函数的神经元。径向基神经网络具有较好的函数逼近能力、分类学习能力和较快的收敛速度。由于径向基神经网络具有性能优良的特点, 近年来也成为神经网络领域研究的热点之一。径向基神经网络广泛应用于众多领域, 包括函数逼近、模式识别、数据或图像压缩、非线性系统建模、机械故障的分析和处理等。

16.3.1 径向基神经网络的基本原理

径向基神经网络为前向的三层网络, 其基本思想是使用具有径向基函数的隐含层神经元, 隐含层将输入的低维数据变换到高维空间, 而隐含层到输出层为线性映射关系。径向基神经网络可以把低维空间无法解决的非线性问题映射到高维空间, 在高维空间通过线性的方式解决, 从而使径向基神经网络具有优良的性能。

最常用的径向基函数是高斯核函数, 其基本形式为 $k(\|x-x_d\|)=\exp(-\|x-x_d\|^2/(2\sigma^2))$ 。

其中, x_c 为核函数中心, σ 为函数的宽度, 控制了径向作用范围。

16.3.2 径向基神经网络的实现

在 MATLAB 中利用函数 newrb() 可构建径向基神经网络, 构建的径向基神经网络在训练的过程中会不断地添加隐含层的神经元的个数, 以使网络满足规定的误差要求。函数 newrb() 的调用格式如下。

[net,tr] = newrb(P,T,goal,spread,MN,DF): 参数 P 和 T 分别为网络的输入和输出, 要求的数据大小为变量数 \times 样本数; 参数 goal 为网络训练达到的目标误差, 默认情况下为 0; 参数 spread 为径向基神经网络函数的扩展速度, 默认情况下为 1; 参数 MN 为神经元的最大个数, 默认为 Q; DF 为两次显示之间添加的神经元的个数, 默认为 2。

函数 newrbe() 可用于构建一个准确的径向基神经网络, 利用该函数可生成零误差的网络模型。函数 newrbe() 的调用格式如下。

net = newrbe(P,T,spread): 参数 P 和 T 同函数 newrb() 为网络的输入和输出, 对于网络的输

入 P, 该函数将生成精确的网络模拟获得输出 T; 参数 spread 为径向基神经网络函数的扩展速度, spread 的值越大, 函数的逼近效果越好, 但是 spread 的值过大也会造成精度的问题。

函数 sim() 用于径向基神经网络的预测, 其使用同 BP 神经网络, 这里不再详细展开叙述。下面以一具体的实例演示径向基神经网络的使用。

【例 16.2】径向基神经网络程序演示。

```
x1=-1:0.1:1;
x2=-1:0.1:1;
y=x1.^2+2*x2-2*sin(x1+x2);           %构建待插值的函数
net = newrbf([x1;x2],y,5)              %构建 RBF 网络
x1=-1:0.01:1;
x2=-1:0.01:1;
yp=sim(net,[x1;x2]);                   %生成插值后的函数
yp=sim(net,[x1;x2]);                   %网络预测
scatter(y,yp)                           %绘制散点图观察结果
xlabel('实际值');
ylabel('预测值');
title('RBF 训练效果');
```

执行上述程序, 将生成如图 16.5 所示的 RBF 函数插值效果图, 显示实际值与网络预测值的关系。

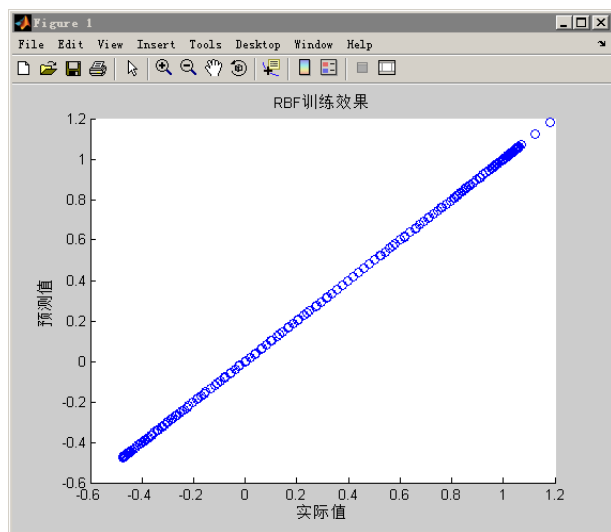


图 16.5 RBF 神经网络的函数插值效果

16.4 广义回归神经网络

广义回归神经网络是基于径向基神经网络, 在其基础上添加一个特殊的线性层构成的网络, 常用于求解函数逼近等问题。GRNN 模型具有良好的性能, 与 BP 等神经网络相比具有较好的收敛性和较高的精度, 而且模型结构简单、计算效率高, 具有良好的应用性。

广义回归神经网络在 MATLAB 中是通过函数 newgrnn() 实现的, 其调用格式如下。

net = newgrnn(P,T,spread): 参数 P 和 T 分别为广义回归神经网络的输入和输出, 参数 spread 为径向基扩展速度, 返回构建的网络 net。

【例 16.3】 广义回归神经网络程序演示。

```

P = [1 2 3 4 5 6 7 8 9 10]; %网络输入参数
T = [2.0 4.1 5.9 7.8 10 12.2 14.1 16 18.3 20.5]; %网络输出参数
net = newgrnn(P,T); %广义回归神经网络构建
P=1.5;
Y = sim(net,P) %网络预测
Y =
    3.3980

```

16.5 自组织神经网络

自组织神经网络是一种非监督的神经网络，即构建网络时无须提供网络的目标输出。自组织神经网络具有良好的自我学习能力，根据样本的特征网络会自动调整权值，以获得目标输出。自组织神经网络适用于解决类别未确定的分类问题，可以自动的对样本分类，且无须确定样本的类别数，避免了用户主观判断对分类结果的影响。

在 MATLAB 实现自组织神经网络的函数为 `newsom()`，其调用格式如下。

`net = newsom(PR,[D1,D2,...],TFCN,DFCN,OLR,OSTEPS,TLR,TND)`：PR 为输入各变量的最大值和最小值，为二维矩阵；[D1,D2,...]为各层的维数，默认为[5 8]；TFCN 为拓扑函数，默认为 'hextop'；DFCN 为距离函数，默认为 'linkdist'；OLR 为分类阶段的学习速率，默认为 0.9；OSTEPS 为分类阶段的步长，默认为 1000；TLR 为调整阶段的速率，默认为 0.02；TND 为调整阶段的领域距离，默认为 1。

【例 16.4】 自组织神经网络程序演示。

```

P = [rand(1,10)*2; rand(1,10)];
net = newsom([0 2; 0 1],[3 5]); %构建 SOM 网络
net=train(net,P); %训练网络
y=sim(net,P); %网络预测
Y=vec2ind(y) %获得网络的分类结果
Y =
    15    14     1    13     4    15     9     8     6    13

```

16.6 神经网络的 GUI 界面实现

在前面的章节中主要介绍了如何通过函数的方式实现神经网络，对于不熟悉 MATLAB 编程，而又仅想利用神经网络的用户来说，MATLAB 神经网络工具箱为广大用户提供了 GUI 界面实现神经网络的方式。通过神经网络 GUI 界面用户无须编写复杂的代码，通过鼠标和键盘操作即可利用界面操作实现网络的训练建模。

神经网络的 GUI 界面具有简洁、友好的用户操作环境，可以方便地构建神经网络，用于数据拟合、模式识别等多方面。下面具体讲述神经网络的 GUI 界面的使用。

1. 神经网络的 GUI 界面的启动

在 MATLAB 命令窗口中输入 `nntool`，即可打开如图 16.6 所示的 Network/Data Manager(神经网络/数据管理)窗口。

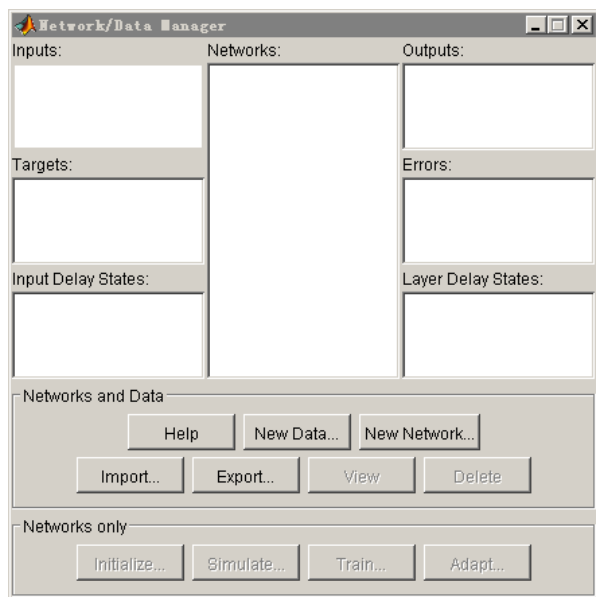


图 16.6 “Network/Data Manager”窗口

2. 利用 GUI 实现神经网络的环境

“Network/Data manager”窗口包含 7 个显示区域和两个按钮区域。其中各显示区域的功能如下。

- “Inputs”区域：显示用户指定的神经网络输入的变量名。
- “Targets”区域：显示用户指定的神经网络目标输出的变量名。
- “Input Delay States”区域：显示用户指定的神经网络输入延迟参数的变量名。
- “Networks”区域：显示用户定义的神经网络名。
- “Outputs”区域：显示神经网络的仿真输出变量名。
- “Errors”区域：显示神经网络的训练误差变量名。
- “Layer Delay States”区域：显示用户指定的神经网络层延迟参数的变量名。

两个按钮区域的各按钮的功能如下。

“Networks and Data”按钮区的功能如下。

- “Help”按钮：用于快速地获取神经网络工具箱的使用帮助文档。单击该按钮，将打开如图 16.7 所示的“Network / Data Manager Help”窗口，为用户提供使用帮助。
- “New Data”按钮：用于构建新的数据。单击该按钮将打开“Create New Data”窗口，如图 16.8 所示，其中在该窗口的“Name”区域可以定义生成的变量的名称，在“Value”区域可以直接输入变量的数据值，而“Data Type”区域指定创建的变量在神经网络中的作用（输入变量（Inputs）、输出变量（Outputs）等），最后单击窗口中的“Create”按钮创建变量。

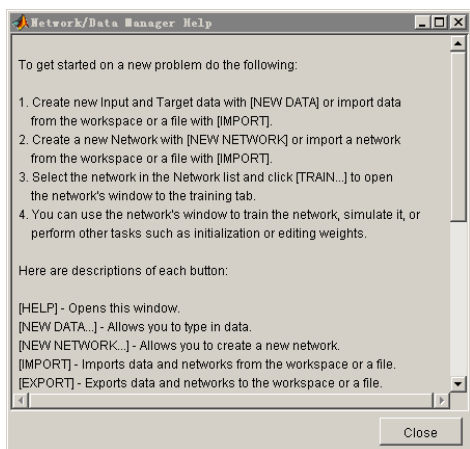


图 16.7 “Network / Data Manager Help” 窗口

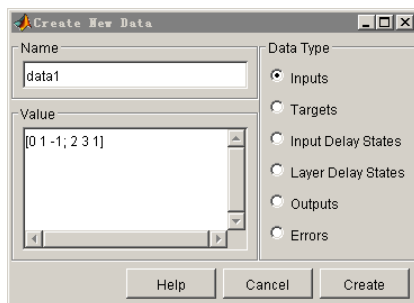


图 16.8 “Create New Data” 窗口

- “New Network” 按钮: 用于创建新的网络。单击该按钮将打开如图 16.9 所示的“Create New Network”窗口, 在“Network Name”区域可以定义神经网络名称; “Network Type”区域定义神经网络类型, 当用户选择了指定的神经网络后, 下方将显示可以设置的网络参数, 例如图 16.9 中选择了“Feed-forward backprop”网络, 下面显示的参数包括 Input ranges (网络输入数据范围)、Training function (训练函数)、Adaption learning function (自适应学习函数)、Performance function (网络表现函数)、Number of layers (网络层数); 对于一般的三层 BP 神经网络此处的网络层数为 2, 即 MATLAB 中只考虑隐含层和输出层的层数, 同时也只需设置隐含层和输出层的参数, 包括 Number of neurons (各层的神经元) 和 Transfer Function (传递函数)。
- “Import..” 按钮: 用于导入神经网络的建模参数。单击该按钮将弹出如图 16.10 所示的“Import or Load to Network/Data Manager”对话框, 通过该对话框可以向神经网络模型中导入数据。对话框中“Source”设置导入数据的来源, 可以来源于 MATLAB 工作空间或者为外部文件, 当选择“Import from MATLAB workspace”单选按钮在对话框的“Select a Variable”区域将显示 MATLAB 工作空间已存的变量, 选中需要导入的变量, 并在“Destination”区域设置导入变量的变量名和在神经网络中变量的作用, 单击“Import”按钮将可向网络导入数据。导入的变量的变量名将显示在“Network/Data manager”窗口的“Inputs”区域。

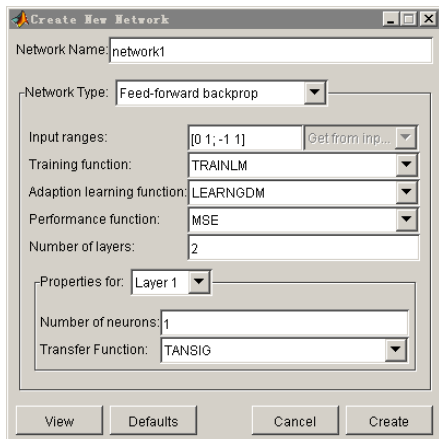


图 16.9 “Create New Network” 窗口

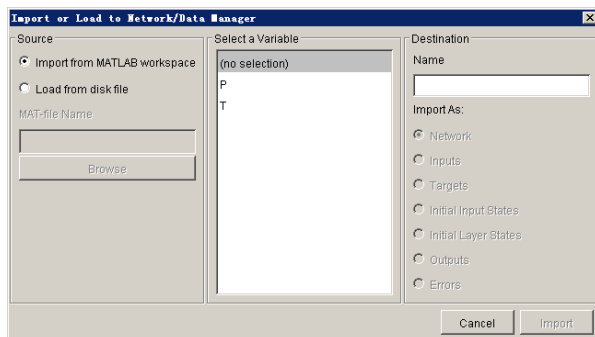


图 16.10 “Import or Load to Network/Data Manager” 对话框

- “Export...” 按钮：用于神经网络训练后产生的仿真结果的导出。单击该按钮将弹出如图 16.11 所示的 “Export or Save from Network/Data Manager” 对话框，可以选择需要导出的神经网络的结果变量，其中 “Select All” 按钮用于选中所有变量导出，“Select None” 按钮用于取消选中。在确定需要导出的变量或网络后单击 “Export” 按钮数据将导入 MATLAB 工作空间中。导出变量的变量名将显示在 “Network/Data manager” 窗口的 “Outputs” 区域。
- “View” 按钮：用于查看神经网络的变量或设置好的网络结构。当在 “Network/Data manager” 窗口选中设置好的网络后，单击此按钮，将打开如图 16.12 所示的网络查看窗口，可以查看网络的具体内容。
- “Delete” 按钮：用于删除网络或其中的变量。在 “Network/Data manager” 窗口选中显示的变量名或网络名，单击此按钮即可删除选中的变量或网络。

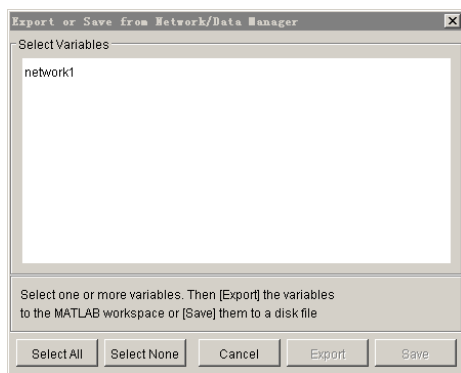


图 16.11 “Export or Save from Network/Data Manager” 对话框

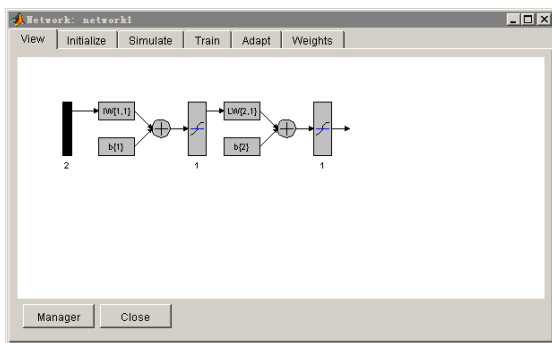


图 16.12 网络查看窗口

“Networks only” 按钮区的功能如下。

- “Initialize” 按钮：用于对网络初始化。
- “Simulate” 按钮：用于对网络仿真。
- “Train” 按钮：用于对网络训练。
- “Adpat” 按钮：用于网络的自适应训练。

3. 利用 GUI 实现神经网络

在前面介绍中读者应该对神经网络 GUI 的启动和基本环境有了详细的了解，下面将以一具体的实例向用户演示如何利用 GUI 实现神经网络。

【例 16.5】神经网络的 GUI 界面实现。

本实例的样本数据如下。

```
P=[-1 -1 2 2; 0 3 0 3];
T=[-1 -1 1 1];
```

在命令窗口输入上述命令，将数据导入 MATLAB 的工作空间中。

(1) 启动神经网络 GUI 界面。在 MATLAB 命令窗口输入 `nntool`, 打开 “Network/Data Manager” 窗口。

(2) 新建网络并设置相应参数。单击 “Network/Data Manager” 窗口中的 “New Network” 按钮新建网络，如图 16.13 所示，其中，

- Network Name (输入网络名): `nettest`。
- Network Type (网络类型): `feed-forward backprop`。
- Input ranges (输入向量的取值范围): `[-1 2; 0 3]`。
- Training function (训练函数): `TRAINLM`。

- Adaption learning function (自适应调整学习函数): LEARNGDM。
- Performance function (误差性能函数): MSE。
- Number of layers (网络层数): 2。
- Properties for(各网络层的属性): Layer1 层 Number of neurons(神经元数) 为 5, Transfer function(传递函数) 为 TANSIG, Layer1 层 Number of neurons(神经元数) 为 1, Transfer function(传递函数) 为 LOGSIG。

(3) 查看网络结构。单击“Network/Data Manager”窗口中的“View”按钮查看新建的网络,如图 16.14 所示。新构建的三层网络包含两个节点的输入层,五个节点的隐含层和一个节点的输出层。

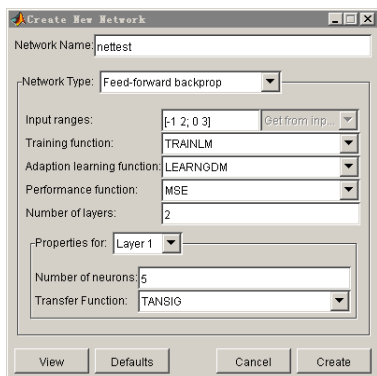


图 16.13 新网络的构建

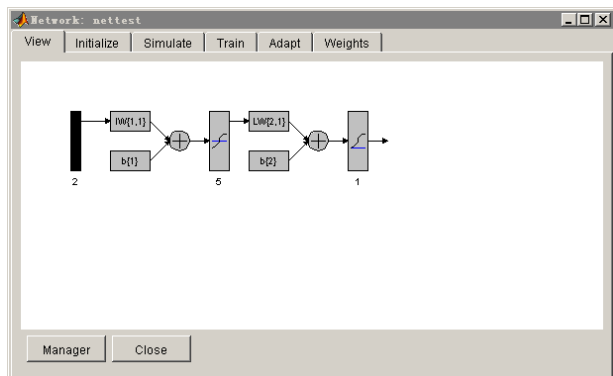


图 16.14 新网络的结构查看

(4) 导入数据,确定网络训练样本的数据。单击“Network/Data Manager”窗口中的“Import data”按钮,在弹出的“Import or Load Network/Data Manager”对话框内,选择从 MATLAB 工作空间导入变量。其中,变量 P 设置为 Name (变量名) P 的 Inputs (输入变量), 变量 T 设置为 Name (变量名) P 的 Targets (输出变量), 如图 16.15 所示。同时,在“Network/Data Manager”窗口中单击“View”按钮可以查看导入数据的具体数值,如图 16.16 所示。

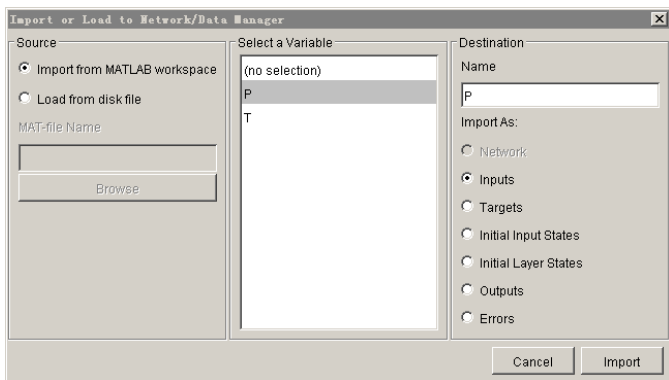


图 16.15 网络数据的导入

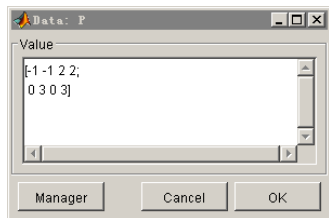


图 16.16 网络数据的查看

(5) 训练网络。在“Network/Data Manager”窗口选中网络 nettest, 单击“Train”按钮, 将打开网络训练参数设置窗口, 在如图 16.17 所示网络信息窗口中设置 Training Data (训练数据) 的 Inputs (输入向量) 为 P, Targets (目标输出数据) 选择 T; Training Results (训练结果) 的 Outputs (输出变量) 为 nettest_output, Errors (误差性能变量) 为 nettest_errors。在训练参数选项卡 (如图 16.18 所示) 中可以设置网络训练时的参数, 用户可以根据实际需要设置不同的参

数，这里采用默认设置。单击“Train Network”按钮开始训练网络，显示如图 16.19 所示的网络训练的误差，网络很快的收敛。

(6) 网络数据的删除和输出。在“Network/Data Manager”窗口将显示网络的输入变量名、目标输出变量名、训练输出变量名和误差名，如图 16.20 所示。此时选中任意数据或网络的名称，单击“Delete”按钮即可删除变量或网络。同时，网络数据的输出通过单击按钮“Export”，可以弹出如图 16.21 所示的数据导出对话框。其中，选中需要导出的数据，单击“Export”按钮后，网络模型的数据将导入到 MATLAB 工作空间中。至此神经网络模型建模工作基本完成。

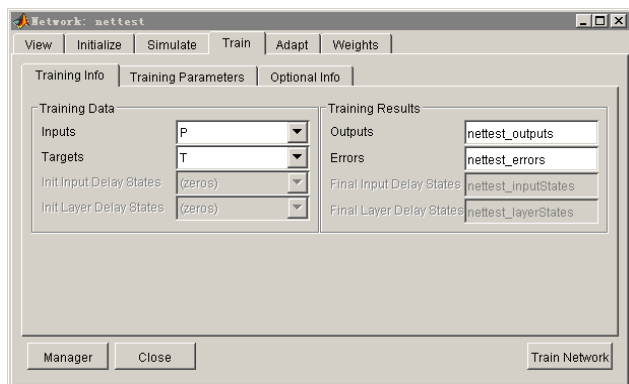


图 16.17 网络训练信息窗口

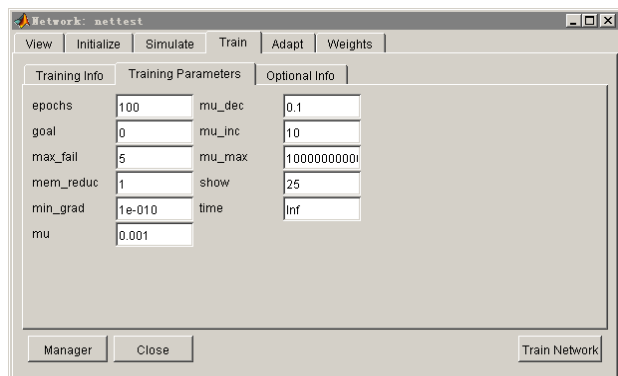


图 16.18 网络训练参数窗口

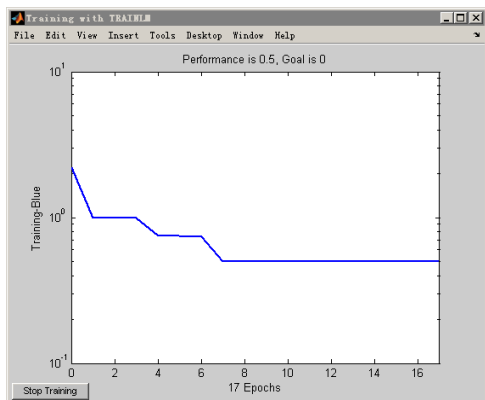


图 16.19 网络训练的误差显示

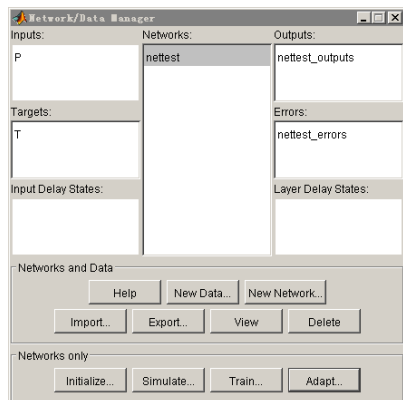


图 16.20 “Network/Data Manager”窗口的数据显示

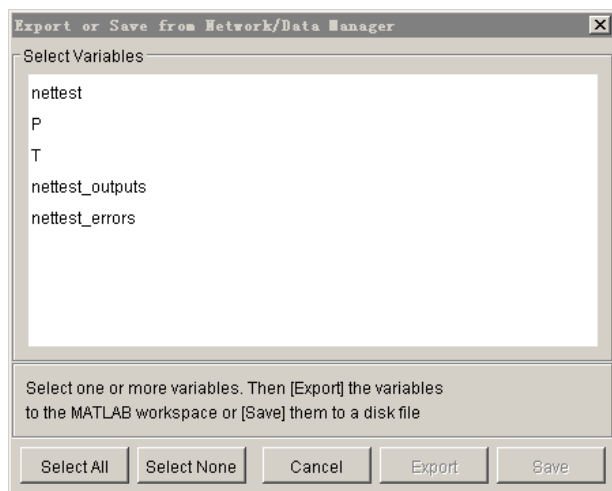


图 16.21 网络数据导出对话框

16.7 本章小结

本章主要介绍了 MATLAB 神经网络工具箱的基本知识。首先介绍了人工神经网络的基础知识，在读者对人工神经网络有了初步的了解后，接着向读者介绍了目前比较常用的神经网络，包括网络的基本原理等，重点放在如何通过 MATLAB 的函数命令实现 BP 神经网络、径向基神经网络、自组织神经网络、广义回归神经网络，其中又以 BP 神经网络的介绍最为详细。最后一节的介绍对于不熟悉 MATLAB 编程的读者更为有用，该节主要介绍了如何利用 GUI 界面方便地实现神经网络。

通过本章的学习读者应该对人工神经网络有基本的了解，熟悉 MATLAB 神经网络工具箱求解神经网络模型的两种方法，在实际应用中读者应该根据实际遇到的问题选择合适的网络，根据自身的能力特点选择适当的求解方法。



第17章

金融工具箱

近年来随着 MATLAB 软件的不断完善,不仅在工程科学领域得到了广泛应用,而且在金融领域也受到了用户的大力推崇,这主要得益于 MATLAB 是一款科学的数据分析软件,可以方便地计算复杂的金融问题,同时其强大的可视化功能,也吸引着无数金融领域的用户使用它。

本章将向读者介绍 MATLAB 金融工具箱的使用,通过本章的介绍希望能进一步推进 MATLAB 在金融计算中的应用。

17.1 金融工具箱简介

在 MATLAB 7.0 中与金融计算相关的工具箱有: Financial Toolbox (金融工具箱)、Financial Derivatives Toolbox (金融衍生产品工具箱)、Financial Time Series Toolbox (金融时间序列工具箱)、Fixed-Income Toolbox (固定收益工具箱)和 Garch Toolbox (广义自回归条件异方差模型工具箱)。下面具体介绍。

1. Financial Toolbox

金融工具箱集成并扩展了 MATLAB 统计和优化工具箱的一些可用于金融计算的函数,使得金融工具箱可以满足大量金融数据的统计分析和可视化。金融工具箱可进行的金融分析囊括了目前常用的金融计算分析功能,包括了投资组合的分析、利率计算、现金流量分析、期权分析等。

2. Financial Derivatives Toolbox

金融衍生产品工具箱主要扩展了金融工具箱中与利率、权益相关的计算,可以使用固定利率、浮动利率等建模方法来计算价格及其演变,提供了多种模型计算基本期权和奇异期权的价格及其演变,并对价格的变动进行灵敏度分析。

3. Financial Time Series Toolbox

金融时间序列工具箱用于对金融市场的时序数据进行分析。金融领域的很多数据是按照时间波动的数据序列,例如股票每天的价格走势等,金融时间序列工具箱为用户提供了更为方便地处理此类数据的工具函数。

4. Fixed-Income Toolbox

固定收益工具箱可用于对固定收益进行建模和分析。工具箱包括对公司债券、国库券、政府债券、存款单和国债等多种固定收益的定价、估价、分析。工具箱主要通过参数拟合和自助法对金融数据进行收益曲线的拟合。

5. Garch Toolbox

广义自回归条件异方差模型工具箱提供了 GARCH 模型进行金融数据分析,主要针对波动性强的单变量的时序特征建模。工具箱提供了对金融数据的广义自回归条件异方差模型构建, Monte

Carlo 模拟和时间序列的预估计和趋势估计分析。

上述工具箱基本包括了常见的各种金融计算功能，适用于常规的金融数据分析。同时在金融数据分析中如果需要使用 MATLAB 其他工具箱的函数也是可以直接使用的，MATLAB 软件的强大性使金融数据可以采用更多的数据分析方法来分析。

17.2 金融数据的获取和可视化

通过前面的介绍，读者对 MATLAB 金融工具箱的组成和基本的功能有了简单的了解，而在开始介绍 MATLAB 金融工具箱的各种功能之前，将向读者详细地介绍金融数据的获取和可视化，因为如果无法准确地获取金融数据导入 MATLAB 中，在 MATLAB 中的金融分析都将是无意义的。同时对于金融数据的可视化可以便于我们更好地了解数据的特征。

金融数据与一般数据相比的最大特征就是其往往带有时序特征，在金融数据中包含时间变量，因而本章将重点讲述一些时间数据在 MATLAB 中的获取和可视化，其他数据的导入/导出和可视化读者可以参看本书专门的数据导入/导出和绘图章节。同时在本节中对于金融数据的可视化，我们将介绍一些专门用于金融领域的曲线的绘制方法。

17.2.1 金融数据的获取

本节主要讲述金融数据导入的相关知识，其中时间数据的导入与之前章节中介绍的 MATLAB 的数据导入机制有所差异。本节金融数据的获取将以时间数据获取的讲解为主，其他的常规数据的导入，请读者自行参考本书其他章节的相关介绍。

MATLAB 接受常规日期时间数据的各种格式的导入，但是导入的日期数据只是字符串形式，需要转换为数值的形式。其转换的机制认为公元元年 1 月 1 日 0 点 0 分为数值 0，距离此时间的天数将以每一天代表数值 1 来转换。例如 2011 年 6 月 15 日距离公元元年 1 月 1 日 734669 天，那么 2011 年 6 月 15 日在 MATLAB 内部可以识别的机制中为数值 734669。

函数 `datenum()` 用于将读入的常规日期数据转换为 MATLAB 可以处理的数值型的日期数据，其调用格式如下。

- $N = \text{datenum}(V)$: 输入数据 V 为常规日期数据格式，可以为具有按年、月、日、时、分、秒顺序排列的六列矩阵数据，或者为按年、月、日顺序排列的三列矩阵数据。
- $N = \text{datenum}(S, F)$: 转换具有一定格式的日期数据 S , S 的数据类型为字符串型或元胞数组， S 为一维的元素，且其中各日期变量需具有相同的格式， F 用于指定输入日期的格式，参看表 17.1，其中列出了 MATLAB 支持的标准日期格式的写法，表中的“字符串表示”即指定的日期数据的格式。
- $N = \text{datenum}(S, F, P)$: 按照指定格式 F 转换日期数据 S ，并指定基准年份为 P ，基准年用于对日期数据 S 中年份只使用两位整数表示的场合，基准年份用于确定这个两位整数年份的前两位，例如基准年份为 1400， S 中年份位置上 10 即表示 1410 年。
- $N = \text{datenum}(Y, M, D)$: 输入数据 Y 、 M 和 D 分别为年、月和日数据的向量，需要具有相同大小。
- $N = \text{datenum}(Y, M, D, H, MI, S)$: 输入数据 Y 、 M 、 D 、 H 、 MI 、 S 分别为年、月、日、时、分、秒数据的向量，需要具有相同大小。
- $N = \text{datenum}(S)$: 导入具有表 17.1 所示 MATLAB 标准的日期格式中编号为 0、1、2、6、13、14、15、16 或者 23 的字符串格式的日期数据 S 。

- `N = datenum(S, P)`: 导入日期数据 `S` 并指定基准年 `P`。

表 17.1 MATLAB 标准的日期格式

编 号	字符串表示	举 例
0	'dd-mmm-yyyy HH:MM:SS'	01-Mar-2010 13:45:15
1	'dd-mmm-yyyy'	1-Mar-2010
2	'mm/dd/yy'	03/01/00
3	'mmm'	Mar
4	'm'	M
5	'mm'	03
6	'mm/dd'	03/01
7	'dd'	01
8	'ddd'	Wed
9	'd'	W
10	'yyyy'	2010
11	'yy'	10
12	'mmyyy'	Mar10
13	'HH:MM:SS'	13:45:15
14	'HH:MM:SS PM'	3:45:17PM
15	'HH:MM'	15:45
16	'HH:MM PM'	3:45PM
17	'QQ-YY'	Q1-96
18	'QQ'	Q1
19	'dd/mm'	01/03
20	'dd/mm/yy'	01/03/00
21	'mmm.dd,yyyy HH:MM:SS'	Mar.01,2010 15:45:17
22	'mmm.dd,yyyy'	Mar.01,2010
23	'mm/dd/yyyy'	03/01/2010
24	'dd/mm/yyyy'	01/03/2010
25	'yy/mm/dd'	2010/03/01
26	'yyyy/mm/dd'	2010/03/01
27	'QQ-YYYY'	Q1-1996
28	'mmyyyyy'	Mar-2010
29	(ISO 8601)'yyyy-mm-dd'	2000/03/01
30	(ISO 8601)'yyyymmdd THHMMSS'	20000301T154517
31	'yyyy-mm-dd HH:MM:SS'	2010-03-01 15:45:17

【例 17.1】利用函数 `datenum()` 转换日期数据。

```
>> date1=[2010 1 1;2010 1 2;2010 1 3]; %导入日期数据的矩阵形式，数据为年、月、日的三列数据
datenum(date1) %转换年、月、日格式的日期数据到数值型的数据
ans =
    734139
    734140
    734141
>> datenum('19-May-2010', 'dd-mmm-yyyy') %指定格式的数据转换
ans =
    734277
>> n = datenum('19-May-10', 'dd-mmm-yy', 2000) %导入指定格式的日期数据，并指定其基准年份
n =
    734277
>> n = datenum(2010, 5, 19) %转换具有年、月、日数据的日期数据
n =
    734277
>> n = datenum([2010 5 19 18 0 0]) %转换具有年、月、日、时、分、秒数据的日期数据
```

```
n =
    7.342777500000000e+005
```

Excel 是目前用户使用比较多的数据存储的载体,下面的一个实例将介绍如何将 Excel 中的日期型的数据导入 MATLAB 并转换为数值型的日期数据。

【例 17.2】 Excel 中日期数据的导入和转换。

现有某股票 10 天的开盘价和收盘价的数据,存储在 Excel 文件中,具体数据格式如表 17.2 所示。

表 17.2 股票 10 天的开盘价和收盘价的数据

日 期	开盘价	收盘价
2010/5/1	5.62	5.74
2010/5/2	5.77	5.78
2010/5/3	5.78	6.2
2010/5/4	6.23	6.67
2010/5/5	6.67	6.56
2010/5/6	6.55	6.34
2010/5/7	6.34	6.55
2010/5/8	6.55	6.66
2010/5/9	6.68	6.9
2010/5/10	7.23	7.43

```
[data,str]=xlsread('test1.xls') %Excel 中的导入数据,其中 data 中存储了数据变量, str 中存储了字符串变量
```

```
datenum(str(2:end,1), 'yyyy/mm/dd') %获取字符串变量中的日期变量,并按指定的格式转换为数值型的日期变量
```

```
ans =
```

```

    734259
    734260
    734261
    734262
    734263
    734264
    734265
    734266
    734267
    734268
```

数值型的时间变量可以方便 MATLAB 处理,当 MATLAB 完成金融分析后,一些结果需要返回给用户,为了便于用户观察数据,我们需要把数值型的日期变量转换为一般的日期变量。函数 `datestr()` 用于将数值型或向量型的日期变量转换为字符串型的日期变量,其调用格式如下。

- `S = datestr(V)`: 将向量型的日期变量转换为字符串型的日期变量,向量型日期变量的一般格式为[年 月 日 时 分 秒]。
- `S = datestr(N)`: 将具有连续型数值特征的日期变量转换为字符串型的日期变量。
- `S = datestr(D, F)`: 以指定的日期格式转换向量型、连续数值型、字符串型的日期变量。
- `S = datestr(S1, F, P)`: 根据基准年 P 按照指定的数据格式 F 转换连续型日期变量 S1 到字符串型日期变量。

【例 17.3】 连续数值型日期变量到字符串型日期变量的转换。


```
>> d=datenum('24.01.2010', 'dd.mm.yyyy')
datestr(d)           %将连续数值型日期变量转换到字符串型日期变量
ans =
24-Jan-2010
>> datestr(d, 2)      %将连续数值型日期变量转换到字符串型日期变量，并指定转换的类型
ans =
01/24/10
```

在 MATLAB 中的日期变量除了可以用字符串型和连续整数型表示外，还可以通过向量的形式表示，可以为同时包含年、月、日、时、分、秒或者仅包含年、月、日的向量。例如 2010 年 8 月 26 日 12 点 05 分 0 秒用向量表示为[2010 8 26 12 5 0]。函数 datevec()可用于将连续数值型日期变量和字符串型日期变量转换为向量型的日期变量，函数的调用格式如下。

- $V = \text{datevec}(N)$: 将连续型日期变量 N 转换为包含年、月、日、时、分、秒的向量型的日期变量。
- $V = \text{datevec}(S, F)$: 转换字符型日期变量 S 到向量型日期变量 V ，需指定转换的字符串变量的格式为 F 。
- $V = \text{datevec}(S, F, P)$: 按照指定格式 F 转换字符串日期变量 S ，并以 P 为基准年。
- $[Y, M, D, H, MI, S] = \text{datevec}(\dots)$: 转换日期变量到向量形式，并使年、月、日、时、分、秒分别保存到变量 Y 、 M 、 D 、 H 、 MI 和 S 中。
- $V = \text{datevec}(S)$: 转换字符串型的时间变量到向量型的日期变量，其中字符串型的时间变量需具有表 17.1 所示 MATLAB 标准的日期格式中编号为 0、1、2、6、13、14、15、16 或者 23 的字符串格式。

【例 17.4】连续数值型日期变量到字符串型日期变量的转换。

```
>> d=datenum('24.01.2010', 'dd.mm.yyyy');
T=datevec(d)           %转换连续数值型的日期变量到向量型的日期变量
T =
Columns 1 through 5
    2010         1        24         0         0
Column 6
         0
>> T=datevec('12/24/2010 11:45')      %转换字符串型的日期变量到向量型的日期变量
T =
Columns 1 through 5
    2010        12        24        11        45
Column 6
         0
```

另外还有一些函数与时间变量的操作有关，在此仅做简单介绍，便于读者对 MATLAB 时间变量的综合应用。

1. 函数 clock()

函数 clock()可用于获取当前计算机的时间，并返回到向量形式的日期变量中，其调用格式如下。

- $c = \text{clock}$: 返回以向量形式存在的系统时间变量。
- $\text{fix}(\text{clock})$: 返回整数形式的日期向量。

2. 函数 now()

函数 now()也可获取当前系统的时间，获取的时间变量是以双精度形式的连续数值变量存在，其调用格式如下。

- $t = \text{now}$: 以连续数值形式返回当前的时间变量。
- $\text{rem}(\text{now}, 1)$: 返回时间变量的时间部分的连续数值表示。
- $\text{floor}(\text{now})$: 返回时间变量的日期部分的连续数值表示。

3. 函数 date()

函数 date()以“dd-mmm-yyyy”的字符串形式返回当前的时间变量，其调用格式如下。

str = date: 字符串形式返回当前的时间变量。

4. 函数 weekday()

函数 weekday()可用于返回一个日期变量的星期几和一星期中的第几天，其调用格式如下。

[N, S] = weekday(D): 返回日期变量 D 为星期 S 和一星期中的第 N 天，其中星期天为一星期中的第一天。

5. 函数 eomday()

函数 eomday()用于返回某年某月的最后一天，其调用格式如下。

E = eomday(Y,M): 返回 Y 年 M 月的最后一天。

6. 函数 calendar()

函数 calendar()可用于返回某年某月的日历，其调用格式如下。

- c = calendar: 可用于返回当前月份的日历。
- c = calendar(d): 可用于返回连续数值变量 d 所在的月份的日历。
- c = calendar(y,m): 可用于返回 y 年 m 月的日历。

【例 17.5】其他时间相关函数的应用。

```
%系统当前时间的获取
>> T1 = clock
T1 =
    1.0e+003 *
    Columns 1 through 3
    2.011000000000000    0.006000000000000    0.015000000000000
    Columns 4 through 6
    0.020000000000000    0.022000000000000    0.025746000000000
>> T2 = fix(clock)
T2 =
    Columns 1 through 5
    2011         6         15         20         22
    Column 6
    28
>> T1=now
T1 =
    7.346698586367014e+005
>> T2=rem(now,1)
T2 =
    0.85866012726910
>> T3=floor(now)
T3 =
    734669
>> str=date
str =
15-Jun-2011
>> [d,w]=weekday('15-Jun-2011')    %返回日期变量的星期几和一星期中的第几天
d =
    4
w =
Wed
%日期查询
>> E = eomday(2010,2)
E =
    28
>> calendar
Jun 2011
```

S	M	Tu	W	Th	F	S
0	0	0	1	2	3	4
5	6	7	8	9	10	11
12	13	14	15	16	17	18
19	20	21	22	23	24	25
26	27	28	29	30	0	0
0	0	0	0	0	0	0

17.2.2 金融数据的可视化

本小节将主要讲述金融数据的可视化。数据的可视化是 MATLAB 的强项，在前面的章节中读者可以看到利用 MATLAB 可以绘制各种各样的图形，用于表述数据特征。本小节将以金融领域的特殊图形绘制的讲解为主。

1. Candlestick chart (蜡烛图)

蜡烛图是人们分析期货、股票、外汇等证券市场价格走势的一项重要工具。函数 `candle()` 可用于绘制蜡烛图，其调用格式如下。

`candle(High, Low, Close, Open, Color)`: 参数 `High` 为最高价，`Low` 为最低价，`Close` 为收盘价，`Open` 为开盘价，`Color` 设置蜡烛图烛台的颜色，默认为当前的背景色。

【例 17.6】蜡烛图的绘制。

```
[data, str]=xlsread('test2.xls');           %从外部文件中读取数据
High=data(:,1);                             %获取最高价数据
Low=data(:,2);                              %获取最低价数据
Close=data(:,3);                           %获取收盘价数据
Open=data(:,4);                             %获取开盘价数据
candle(High, Low, Close, Open, 'red')       %指定蜡烛图的颜色为红色
```

执行上述程序，将生成如图 17.1 所示的图形。

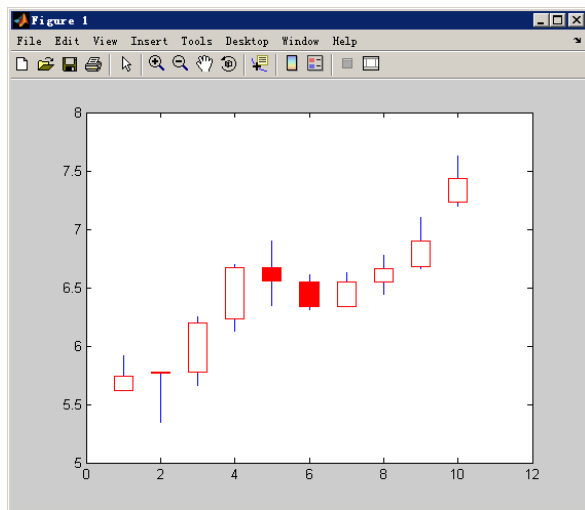


图 17.1 蜡烛图的绘制

2. 时间坐标轴的绘制

上述绘图的数据为某股票 10 日内的价格走势，因而图 17.1 所示中的横坐标应该为时间变量，函数 `dateaxis()` 可用于添加时间变量的横坐标。函数 `dateaxis()` 的调用格式如下。

`dateaxis(Axis, DateForm, StartDate)`: `Axis` 指定时间变量的坐标轴，默认为 `x` 轴；`DateForm` 指定时间变量的格式，编号 0–17 分别对应 18 种不同的时间变量的格式，具体参考 MATLAB 的帮助文档；参数 `StartDate` 用于指定坐标轴的起始时间。

【例 17.7】时间坐标轴的绘制。

为图 17.1 所示的蜡烛图添加时间坐标轴。

```
dateaxis('x',17, '10/05/01')
```

在例 17.6 代码的基础上执行上述代码，将生成如图 17.2 所示的添加时间横坐标的蜡烛图。

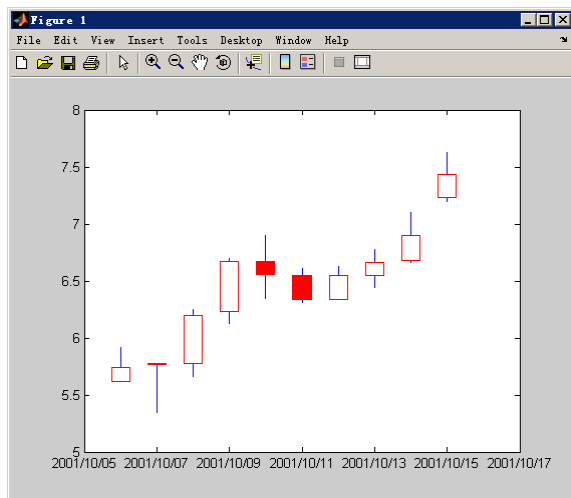


图 17.2 添加时间横坐标的蜡烛图的绘制

3. highlow（高低线图）

高低线图也是反映金融市场价格走势的主要图形。函数 `highlow()` 可用于绘制高低线图，其调用格式如下。

- `highlow(High, Low, Close, Open, Color)`: 参数同蜡烛图。
- `Handles = highlow(High, Low, Close, Open, Color)`: 生成高低线图，并返回句柄。

【例 17.8】高低线图的绘制。

```
[data,str]=xlsread('test2.xls');           %导入数据
High=data(:,1);
Low=data(:,2);
Close=data(:,3);
Open=data(:,4);
highlow(High, Low, Close, Open,'red')      %绘制高低线图
dateaxis('x',17, '10/05/01')
```

运行上述程序，将生成如图 17.3 所示的高低线图。

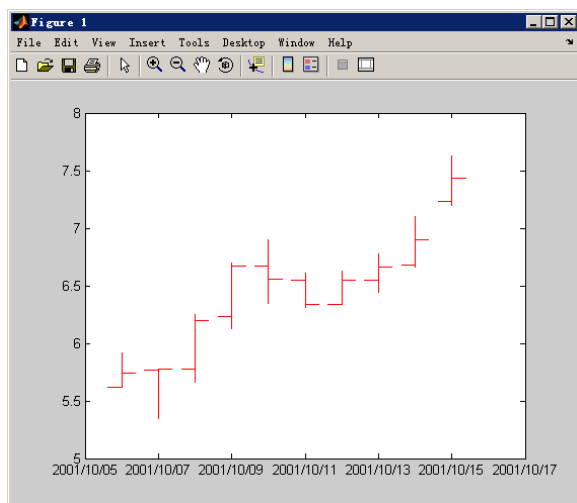


图 17.3 高低线图的绘制

17.3 金融数据分析

本节主要介绍金融数据的常用分析，包括投资组合分析、利率期限计算、资金流量估算和时间序列分析。

17.3.1 投资组合分析

在金融市场中为了在最大可能回避投资风险的同时获取最大的投资回报，人们需要设计合理的投资组合。而为了权衡投资风险与投资回报，目前普遍接受的观点是分散投资组合，分散风险。经济学家哈里·马科维茨提出了衡量资产组合的收益和风险的方法，利用投资的预期收益率、方差、协方差矩阵得到，分析投资组合的有效边界和无差异曲线，从而确定最佳的投资组合。

在 MATLAB 中计算投资的回报和风险的函数为 `portstats()`，其调用格式如下。

`[PortRisk, PortReturn] = portstats(ExpReturn, ExpCovariance, PortWts)`：其中参数 `ExpReturn` 为每项投资的预期收益率；`ExpCovariance` 为投资组合的协方差矩阵；可选参数 `PortWts` 为资产组合的权值矩阵。

【例 17.9】投资的回报和风险。

现有投资项目 A、B 和 C，其预期的收益率分别为 20%、30% 和 15%，投资组合的协方差矩阵如下。

```
0.0100, -0.0061, 0.0042
-0.0061, 0.0400, -0.0252
0.0042, -0.0252, 0.0225
```

投资组合的权值为 0.3、0.4 和 0.3。

```
>> ExpReturn = [0.2 0.3 0.15];
ExpCovariance = [0.0100 -0.0061 0.0042
                 -0.0061 0.0400 -0.0252
                 0.0042 -0.0252 0.0225];
PortWts=[0.3 0.4 0.3];
[PortRisk, PortReturn] = portstats(ExpReturn, ExpCovariance, PortWts) %计算投资的回报
和风险
PortRisk =
0.05068530358990
```

```
PortReturn =
0.225000000000000
```

17.3.2 利率期限计算

利率可用于反映金融投资的回报情况,对于在投资时固定利率的债券或者固定存款存在利率的风险,即利率变化导致投资回报的变化。因而,在金融行为中研究利率期限的变换规律是很有必要的,可以帮助我们回避利率变化而造成的损失。MATLAB 中计算利率期限的方法为 bootstrapping 方法,其实现的函数为 `zbtyield()` 和 `zbtprice()`,下面具体介绍这两个函数的使用。

函数 `zbtyield()` 通过债券收益率计算利率期限,其调用格式如下。

`[ZeroRates, CurveDates] = zbtyield(Bonds, Yields, Settle, OutputCompounding)`: 参数 `Bonds` 包含了用于产生债券利率期限曲线的数据,为 $n \times 2$ 至 $n \times 6$ 的数据矩阵,包含的必须参数有 `Maturity` (债券到期日)、`CouponRate` (以小数形式存在的债券票面利率),可选参数有 `Face` (债券面值)、`Period` (每年债券的付息次数,可以为整数 0、1、2 (默认)、3、4、6 和 12)、`Basis` (计息天数)、`EndMonthRule` (月末法则);参数 `Yields` 为债券收益率;参数 `Settle` 为债券的结算日期;参数 `OutputCompounding` 为计息次数,函数返回零息票利率 `ZeroRates` 和对应的时间 `CurveDates`。

函数 `zbtprice()` 通过债券价格计算利率期限,其调用格式如下。

`[ZeroRates, CurveDates] = zbtprice(Bonds, Prices, Settle, OutputCompounding)`: 大多数参数基本同函数 `zbtyield()`,参数 `Prices` 为债券价格。

【例 17.10】利率期限的计算。

```
Maturity=['1/1/1998';'1/1/1999';'1/1/2000';'1/1/2001';'1/1/2002';'1/1/2003'; ...
'1/1/2004';'1/1/2005';'1/1/2006';'1/1/2007';'1/1/2008';'1/1/2009';'1/1/2010'];
Maturity=datenum(Maturity);
CouponRate=[0.04;0.037;0.035;0.04;0.042;0.052;0.067;0.064;0.068;0.069;0.072;0.065;0.067];
Bonds=[Maturity CouponRate];
Price=[100.3;101.2;102.3;98.89;
99.7;100.23;101.2;102.4;103.12;100.78;103.4;104.2;105.5];
Settle=datenum('6/1/1997')
[ZeroRates, CurveDates] = zbtprice(Bonds, Price, Settle) %利率期限计算
plot(CurveDates,ZeroRates) %绘制利率期限的变化图
dateaxis('x')
ZeroRates =
0.03471462378880
0.02909734849851
0.02558600284087
0.04423638726251
0.04337855754680
0.05338010386294
0.06987847269727
0.06296532636071
0.06707866512293
0.07329528327839
0.07234231619110
0.06038975772092
0.06151677777832
CurveDates =
729756
730121
730486
730852
731217
731582
731947
732313
732678
733043
733408
```

733774

734139

执行上述程序，将计算得到利率期限结构，并做图展示利率期限的变化，如图 17.4 所示。

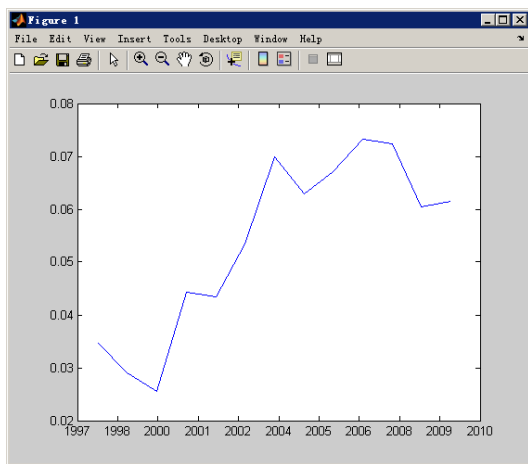


图 17.4 利率期限的结构图

17.3.3 资金流量估算

在金融领域的资金具有明显的时间特性，因而其资金流量的估算需要考虑当前价值和未来价值。本小节主要讨论如何在 MATLAB 中对资金流量估算。

1. 固定现金流的资金估算

在一些金融场合，我们的资金将以固定的周期存入或者取出，例如国债等金融产品，具有固定的收益率，用于以固定的周期把钱存入或去除。函数 `PresentVal()` 可用于固定现金流的当前资金估算，其调用格式如下。

`PresentVal = pvfix(Rate, NumPeriods, Payment, ExtraPayment, Due)`: 参数 `Rate` 为定期利率，使用小数表示；`NumPeriods` 为计息的现金流周期数；`Payment` 为固定的现金流（利息），可选参数 `ExtraPayment` 为最后一次周期的现金流（本金），默认为 0；可选参数 `Due` 为现金流的计息方式，0 代表周期末计息（默认），1 代表周期初计息，函数返回固定现金流的当前资金估算值。

【例 17.11】 现有 5 年期的固定国债 10000 元，利率为 3.8%，在每月的月末投资者取出其固定的收益，5 年后收回本金，其固定现金流的当前估算方式如下。

```
>> Rate=0.038;           %利率
NumPeriods=5;           %计息周期数
Payment=Rate*10000;      %每年的固定收益
ExtraPayment=10000;      %到期本金
PresentVal = pvfix(Rate, NumPeriods, Payment, ExtraPayment) %固定现金流的当前资金估算
PresentVal =
    9.999999999999998e+003
```

函数 `FutureVal()` 可用于固定现金流的未来资金估算（资金终值），其调用格式如下。

`FutureVal = fvfix(Rate, NumPeriods, Payment, PresentVal, Due)`: 参数 `Rate`、`NumPeriods`、`Payment`、`Due` 同函数 `PresentVal()`，参数 `PresentVal` 为现金流的初始值，返回固定现金流的终值 `FutureVal`。

【例 17.12】 按揭还贷，首付 30 万，每月支付 5000，支付 10 年，利率为 5.6%，现金流的终值计算如下。

```
>> Rate=0.056/12;        %利率
NumPeriods=12*10;        %还款周期数
```

```

Payment=5000;           %周期还款额
PresentVal=300000;       %本金
FutureVal = fvfix(Rate, NumPeriods, Payment, PresentVal)%固定现金流的终值计算
FutureVal =
    1.326368271798690e+006

```

2. 变化现金流的资金估算

在一些投资项目中，投资者投入的现金流往往并不是以固定的投入或取出的，对于这些变化现金流的资金估算问题 MATLAB 金融工具箱也提供了函数，可以直接计算变化现金流的当前值和终值。

函数 `PresentVal()` 用于估算当前变化现金流，其调用格式如下。

`PresentVal = pvvar(CashFlow, Rate, IrrCFDates)`: 参数 `CashFlow` 为变化的现金流向量，初始值为投入的资金数，为负值；参数 `Rate` 为以小数点形式表示的周期性的收益率；可选参数 `IrrCFDates` 用于设置现金流的周期，可为非等间隔周期，默认情况下为等间隔的周期；函数返回变化现金流的当前值 `PresentVal`。

函数 `PresentVal()` 用于估算变化现金流的终值，其调用格式如下。

`FutureVal = fvvar(CashFlow, Rate, IrrCFDates)`: 其中各参数的使用同函数 `PresentVal()`，函数返回变化现金流的终值。

【例 17.13】现有一投资，初期投入 10 万，之后五年每年的收益为 2000、3000、5000、10000、12000，收益率为 0.8%，计算变化现金流的当前净值和未来终值。

```

>> CashFlow=[-10000,2000,3000,5000,10000,12000];
PresentVal = pvvar(CashFlow, 0.08)
PresentVal =
    1.391032641022762e+004
>> FutureVal = fvvar(CashFlow, 0.08)
FutureVal =
    2.043883315200000e+004

```

17.3.4 时间序列分析

在前面的内容中读者可以发现金融数据往往带有时间特征，因而时间序列的分析方法适用于金融数据的分析。本小节将主要介绍 MATLAB 时间序列分析的常用函数，最后以一简单实例演示时间序列分析的进行。

1. 函数 `autocorr()`

函数 `autocorr()` 用于计算时间序列的自相关系数，可以反映序列中数据随时间变化的自相关程度，其函数的调用格式如下。

- `autocorr(Series,nLags,M,nSTDs)`: 其中参数 `Series` 为需要计算自相关系数的时间序列，可选参数 `nLags` 为自相关函数的阶数，默认情况下为 0, 1, 2, ..., T, 其中 T 等于 $\min([20, \text{length}(\text{Series})-1])$ ，可选参数 `M` 为当自相关函数的阶数大于 M 的时候，自相关值为 0，可选参数 `nSTDs` 为自相关函数估计误差的标准差。
- `[ACF,Lags,Bounds] = autocorr(Series,nLags,M,nSTDs)`: 计算时间序列的自相关函数，并返回参数 ACF 自相关系数，Lags 与自相关系数对应的阶数，Bounds 自相关系数的置信区间。

2. 函数 `parcorr()`

函数 `parcorr()` 用于计算时间序列的偏相关系数，其函数的调用格式如下。

`[PartialACF,Lags,Bounds] = parcorr(Series,nLags,R,nSTDs)`: 其基本的输入/输出参数同自相关函数 `autocorr()`，返回的 `PartialACF` 为计算得到的偏相关系数。

3. 函数 ar()

函数 ar() 用于估计 AR 模型的参数，其调用格式如下。

$m = \text{ar}(y, n)$: n 为模型的阶数， y 为时间序列， m 为估计的 AR 模型的系数。

4. 函数 armax()

函数 armax() 可用于估计时间序列 ARMAX 或 ARMA 模型的参数，其调用格式如下。

$m = \text{armax}(\text{data}, \text{orders})$: data 为时间序列数据，参数 orders 为确定的时间序列的阶数，返回时间序列模型的系数 m 。

5. 函数 predict()

函数 predict() 可对时间序列模型进行预测，其调用格式如下。

$yp = \text{predict}(m, \text{data})$: 其中参数 data 为待预测的时间序列数据， m 为时间序列的系数，返回时间序列模型的预测值 yp 。

【例 17.14】 现有某零售额 24 年变化的时间序列数据，试构建其时间序列模型。

```
load data;           %导入时间序列数据
autocorr(x)          %模型自相关系数的计算
parcorr(x)           %模型自偏关系数的计算
m=ar(x,12);          %模型参数估计
yp=predict(m,x);     %模型预测
```

执行上述程序，将分别生成如图 17.5 所示的时间序列的自相关图和如图 17.6 所示偏相关图，通过此可基本确定模型的阶数。

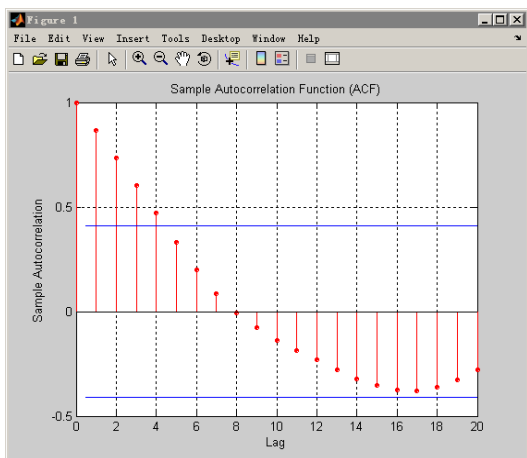


图 17.5 时间序列的自相关图

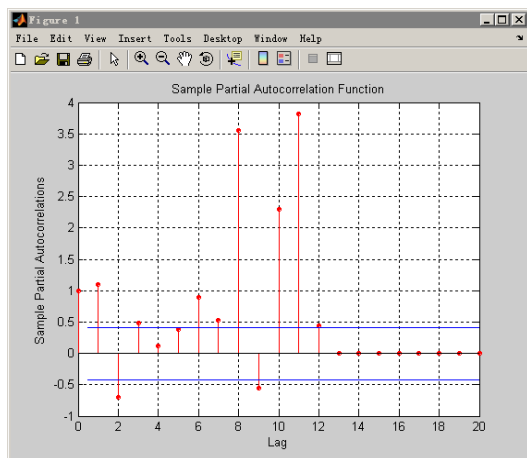


图 17.6 时间序列的偏相关图

17.4 本章小结

本章主要介绍了 MATLAB 金融工具箱的使用。首先简单介绍了 MATLAB 金融工具箱的组成，使读者对金融工具箱有初步的认识，接着，讲述了金融数据的获取和可视化，其中主要集中讲述了时间数据的导入和转换，并演示了金融工具箱特有的一些绘图功能，然后，介绍了如何利用金融工具箱提供的函数进行常规的金融计算，包括投资组合分析、利率期限计算、资金流量估算和时间序列分析，但是限于文章篇幅和笔者的能力，仅介绍了部分的金融数据分析功能，MATLAB 金融工具箱强大的金融数据处理能力还有待用户不断的尝试与探索。

第18章

小波分析工具箱

MATLAB 小波分析工具箱为用户提供了一个功能强大、操作简单的小波分析工具，是一个很好的工程设计、仿真模拟和应用平台。小波分析在信号和图像的处理、分析、去噪、压缩等领域具有广泛的应用。通过本章的学习将使读者掌握小波分析工具箱的初步使用，主要包括以下内容：

- 小波变换的基础知识。
- 常用的小波分析操作。
- 利用 GUI 实现小波分析。

在 MATLAB 小波分析工具箱中实现小波分析的方法主要有函数法和 GUI 图形界面操作法，其中函数法可以按照用户的需要灵活地编写可以重复利用的小波分析代码，而图形界面对于不擅长编写代码的用户来说，使用更为方便和直观。

18.1 小波变换的基础知识

小波变换是一种新的数据信号分析方法，它继承和发展了傅里叶变换的思想，同时又克服了傅里叶变换的窗口大小不随频率变化、无法获取信号的时间信息等缺点，可以通过时频分析的方法对信号进行分析处理，是进行信号分析和处理的理想工具。

小波变换是在母小波函数的基础上，通过对小波函数的平移、伸缩实现对信号的时频分析。小波基函数的基本形式为：

$$\psi_{a,b}(t) = \frac{1}{\sqrt{|a|}} \psi\left(\frac{t-b}{a}\right)$$

其中， a 为尺度参数，用于控制伸缩， b 为平移参数，用于控制位置。

常用的小波函数有 Haar 小波、高斯小波、Morlet 小波和墨西哥帽小波等，需要根据用户实际的信号类型选择合适的小波母函数类型，在 MATLAB 中用户可以利用函数 `waveinfo()` 查阅相关的小波母函数的信息，其调用格式如下。

- `waveinfo`：查看 MATLAB 小波工具箱内的所有小波函数的信息。
- `waveinfo('wname')`：查看名称为 `wname` 的小波函数的信息。

【例 18.1】小波函数信息的获取。

```
>> waveinfo %查看 MATLAB 小波工具箱内的所有小波函数的信息
INFOWAVE Information on wavelets.
Wavelets
1. Crude wavelets.
Wavelets: gaussian wavelets (gaus), morlet, mexican hat (mexihat).
Properties: only minimal properties
- phi does not exist.
- the analysis is not orthogonal.
- psi is not compactly supported.
```

- the reconstruction property is not insured.
 - Possible analysis:
 - continuous decomposition.
 - Main nice properties: symmetry, ψ has explicit expression.
 - Main difficulties: fast algorithm and reconstruction unavailable.
2. Infinitely regular wavelets.
- Wavelets: meyer (meyr).
- Properties:
 - ϕ exists and the analysis is orthogonal.
 - ψ and ϕ are indefinitely derivable.
 - ψ and ϕ are not compactly supported.
- Possible analysis:
 - continuous transform.
 - discrete transform but with non FIR filters.
- Main nice properties: symmetry, infinite regularity.
- Main difficulties: fast algorithm unavailable.
- Wavelets: discrete Meyer wavelet (dmey).
- Properties:
 - FIR approximation of the Meyer wavelet
- Possible analysis:
 - continuous transform.
 - discrete transform.
3. Orthogonal and compactly supported wavelets.
- Wavelets: Daubechies (dbN), symlets (symN), coiflets (coifN).
- General properties:
 - ϕ exists and the analysis is orthogonal.
 - ψ and ϕ are compactly supported.
 - ψ has a given number of vanishing moments.
- Possible analysis:
 - continuous transform.
 - discrete transform using FWT.
- Main nice properties: support, vanishing moments, FIR filters.
- Main difficulties: poor regularity.
- Specific properties:
 - For dbN : asymmetry
 - For symN : near symmetry
 - For coifN: near symmetry and ϕ as ψ , has also vanishing moments.
4. Biorthogonal and compactly supported wavelet pairs.
- Wavelets: B-splines biorthogonal wavelets (biorNr.Nd and rbioNr.Nd).
- Properties:
 - ϕ functions exist and the analysis is biorthogonal.
 - ψ and ϕ both for decomposition and reconstruction are compactly supported.
 - ϕ and ψ for decomposition have vanishing moments.
 - ψ and ϕ for reconstruction have known regularity.
- Possible analysis:
 - continuous transform.
 - discrete transform using FWT.
- Main nice properties: symmetry with FIR filters, desirable properties for decomposition and reconstruction are split and nice allocation is possible.
- Main difficulties: orthogonality is lost.
5. Complex wavelets.
- Wavelets: Complex Gaussian wavelets (cgauN), complex Morlet wavelets (cmorFb-Fc), complex Shannon wavelets (shanFb-Fc), complex frequency B-spline wavelets (fbspm-Fb-Fc).
- Properties: only minimal properties
 - ϕ does not exist.
 - the analysis is not orthogonal.
 - ψ is not compactly supported.

```

- the reconstruction property is not insured.
Possible analysis:
- complex continuous decomposition.
Main nice properties: symmetry, psi has explicit expression.
Main difficulties: fast algorithm and reconstruction unavailable.

```

小波系数为信号表示成一系列不同尺度和不同位置的小波函数线性组合的系数。小波变换本质为连续的变换，但是在实际使用中信号本身往往是离散的，同时利用计算机模拟也不能得到连续的小波变换，所以我们往往使用离散的小波变换。

18.2 常用的小波分析操作

本节主要介绍常用的小波分析函数，指导用户通过函数法实现小波分析操作。其中包括一维小波分析、二维小波分析和小波包变换等。

18.2.1 一维小波分析

本小节主要介绍一维小波分析，包括一维连续小波变换、一维离散小波变换、一维离散小波变换系数的提取和一维离散小波变换系数的重建。

1. 一维连续小波变换

在 MATLAB 小波分析工具箱中函数 `cwt()` 可用于提取连续小波系数，其调用格式如下。

- `COEFS = cwt(S, SCALES, 'wname')`: 在尺度 `SCALES` 上计算输入向量 `S` 的连续小波系数 `COEFS`，`wname` 为小波函数名，返回的小波系数 `COEFS` 为矩阵，其行号对应小波的不同尺度，列对应信号的采样点。
- `COEFS = cwt(S, SCALES, 'wname', 'plot')`: 输入参数 `plot` 用于控制小波系数图的显示，将绘制颜色图用于表示小波系数，其横坐标为采样点，纵坐标为小波分解的尺度，颜色越深，小波系数越大。
- `COEFS = cwt(S, SCALES, 'wname', PLOTMODE)`: 参数 `PLOTMODE` 控制绘制的小波系数图的颜色模式，包括 `lvl` (scale-by-scale 的着色模式)、`glb` (考虑所有尺度的着色模式)、`abslvl` 或 `lvlabs` (使用系数绝对值的 scale-by-scale 的着色模式) 和 `absglb` 或 `glbabs` (使用系数绝对值的考虑所有尺度的着色模式)。
- `COEFS = cwt(S, SCALES, 'wname', PLOTMODE, XLIM)`: 参数 `XLIM` 用于控制小波系数图 X 轴的范围。

【例 18.2】一维连续小波变换。

```

load vonkoch
x=vonkoch(1:500);
subplot(2,2,1)
plot(x); %绘制原始信号图
title('Original signal');
SCALES=[1:32]
subplot(2,2,2)
COEFS = cwt(x, SCALES, 'sym2', 'plot'); %利用函数 cwt() 提取连续小波系数，并绘制小波系数图
subplot(2,2,3)
COEFS = cwt(x, SCALES, 'sym2', 'lvl'); %提取小波系数，并以 scale-by-scale 的着色模式绘制小波系数图
subplot(2,2,4)
COEFS = cwt(x, SCALES, 'sym2', 'glb'); %提取小波系数，并以尺度的着色模式绘制小波系数图

```

执行上述程序，将生成如图 18.1 所示的原始信号和一维连续小波系数图。

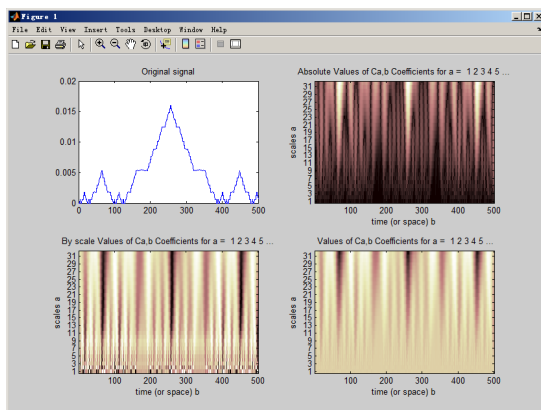


图 18.1 原始信号和一维连续小波系数图

2. 一维离散小波变换

MATLAB 小波工具箱中函数 `dwt()` 和 `wavwdec()` 分别用于实现单尺度、多尺度的一维离散小波变换，下面具体讲述这两个函数的使用。

函数 `dwt()` 可用于单尺度的一维离散小波变换，其调用格式如下。

- `[cA,cD] = dwt(X,'wname')`: 输入参数 `X` 为输入信号，`wname` 为小波函数，返回小波分析后信号的低频系数 `cA` 和低频系数 `cD`。
- `[cA,cD] = dwt(X,'wname','mode',MODE)`: 使用指定的扩展模式 `MODE` 计算一维离散小波系数。
- `[cA,cD] = dwt(X,Lo_D,Hi_D)`: 使用低通滤波器 `Lo_D` 和高通滤波器 `Hi_D` 进行小波变换。
- `[cA,cD] = dwt(X,Lo_D,Hi_D,'mode',MODE)`: 使用低通滤波器 `Lo_D` 和高通滤波器 `Hi_D` 进行扩展模式的小波变换。

【例 18.3】 单尺度的一维离散小波变换。

```
load vonkoch
s=vonkoch(1:500);
[cA1,cD1] = dwt(s,'haar'); %单尺度的一维离散小波变换
subplot(211); plot(s); title('Original signal'); %绘制原始信号
subplot(223); plot(cA1); title('Approx. coef. for haar'); %小波低频系数绘制
subplot(224); plot(cD1); title('Detail coef. for haar'); %小波高频系数绘制
```

执行上述程序，将生成如图 18.2 所示的原始信号和一维离散单尺度小波系数图。

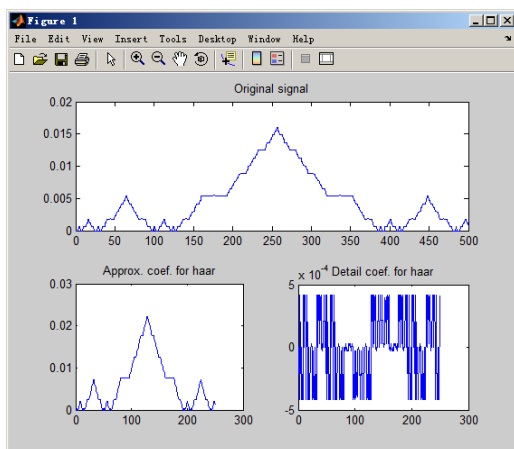


图 18.2 原始信号和一维离散单尺度小波系数图

函数 `wavedec()` 可用于多尺度的一维离散小波变换，其调用格式如下。

- `[C,L] = wavedec(X,N,'wname')`: X 为输入信号，参数 N 为分解的尺度， $wname$ 为小波函数，返回近似分量 C 和细节分量 L 。
- `[C,L] = wavedec(X,N,Lo_D,Hi_D)`: 使用指定的滤波器组 Lo_D 和 Hi_D 对信号进行分解。

【例 18.4】 多尺度的一维离散小波变换。

```
load sumsin;
s=sumsin;
[C,l] = wavedec(s,3,'db1');           %多尺度的一维离散小波变换
subplot(211); plot(s); title('Original signal'); %绘制原始信号
subplot(223); plot(c); title('wavelet decomposition vector C for db1'); %小波近似分量绘制
subplot(224); plot(l); title(' the bookkeeping vector L for db1'); %小波细节分量绘制
```

执行上述程序，将生成如图 18.3 所示的原始信号和一维离散多尺度小波系数图。

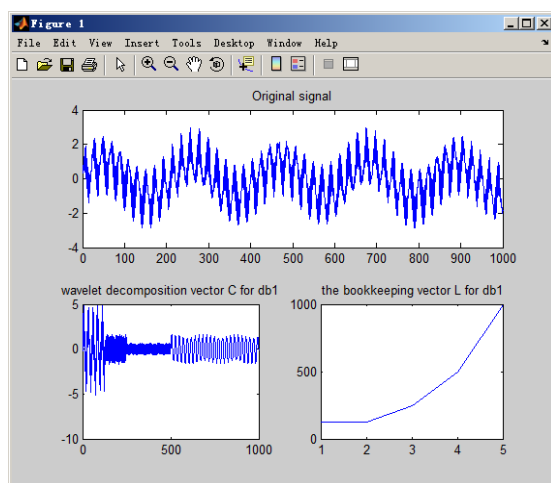


图 18.3 原始信号和一维离散多尺度小波系数图

3. 一维离散小波变换系数的提取

函数 `appcoef()` 和 `detcoef()` 可分别用于提取一维离散多尺度小波变换系数的近似系数和细节系数。函数 `appcoef()` 的调用格式如下。

- `A = appcoef(C,L,'wname',N)`: 输入参数 C 和 L 为多尺度的一维离散小波变换函数 `wavedec()` 的输出参数， $wname$ 为小波函数， N 为小波尺度因子，在 $[0, \text{length}(L)-2]$ 范围内，返回小波变换的近似系数。
- `A = appcoef(C,L,'wname')`: 在尺度 $\text{length}(L)-2$ 上提取一维离散小波变换的近似系数。
- `A = appcoef(C,L,Lo_R,Hi_R)`: 使用低通滤波器和高通滤波器在尺度 $\text{length}(L)-2$ 上提取一维离散小波变换的近似系数。
- `A = appcoef(C,L,Lo_R,Hi_R,N)`: 使用低通滤波器和高通滤波器在尺度 N 上提取一维离散小波变换的近似系数。
- 函数 `detcoef()` 用于提取小波变换的细节系数，其调用格式如下。
- `D = detcoef(C,L,N)`: 输入参数 C 和 L 为多尺度的一维离散小波变换函数 `wavedec()` 的输出参数， N 为小波尺度因子，返回小波变换的细节系数 D 。
- `D = detcoef(C,L)`: 在尺度 $\text{length}(L)-2$ 上提取一维离散小波变换的细节系数 D 。

【例 18.5】 一维离散小波变换系数的提取。

```
load sumsin;
s=sumsin;
[C,l] = wavedec(s,3,'db1');           %一维离散小波变换系数的提取
```

```

A1= appcoef(c,l,'db1',1);
A2= appcoef(c,l,'db1',2);
A3= appcoef(c,l,'db1',3);
%绘制各层的小波近似系数图
subplot(311); plot(A1); title('approximation coefficients at level 1');
subplot(312); plot(A2); title('approximation coefficients at level 2');
subplot(313); plot(A3); title('approximation coefficients at level 3');
[cd1,cd2,cd3] = detcoef(c,l,[1 2 3])
figure;
%绘制各层的小波细节系数图
subplot(311); plot(cd1); title('detail coefficients at level 1');
subplot(312); plot(cd2); title('detail coefficients at level 2');
subplot(313); plot(cd3); title('detail coefficients at level 3');

```

执行上述程序将生成如图 18.4 所示的一维离散小波变换近似系数的提取和图 18.5 所示的一维离散小波变换细节系数的提取。

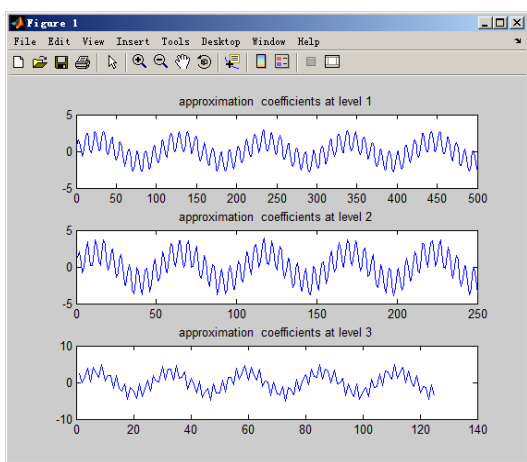


图 18.4 一维离散小波变换近似系数的提取

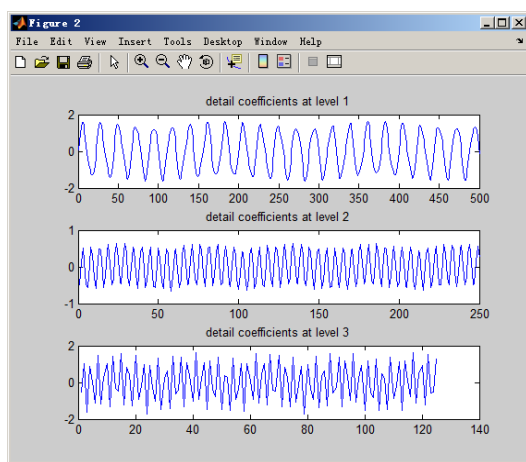


图 18.5 一维离散小波变换细节系数的提取

4. 一维离散小波变换系数的重建

MATLAB 小波工具箱提供了一维离散小波变换系数的重建，包括单尺度和多尺度一维离散小波变换系数的重建。

函数 `idwt()` 可用于一维离散单尺度小波变换系数的重建，其调用格式如下。

- `X = idwt(cA,cD,'wname')`: `wname` 为小波系数，`cA` 和 `cD` 分别为低频系数和高频系数，返回重构的信号 `X`。
- `X = idwt(cA,cD,Lo_R,Hi_R)`: `Lo_R` 和 `Hi_R` 分别为低频和高频滤波器，返回由 `cA` 和 `cD` 分别为低频系数和高频系数重构的系数 `X`。
- `X = idwt(cA,cD,'wname',L)`: 返回信号中心 `L` 个重构的信号点。
- `X = idwt(cA,cD,Lo_R,Hi_R,L)`: 使用滤波器重构信号，并返回信号中心 `L` 个重构的信号点。
- `X = idwt(...,'mode',MODE)`: 使用指定的扩展模式 `MODE` 重构信号。

【例 18.6】 一维离散单尺度小波变换系数的重建。

```

load sumsin;
s=sumsin;
[ca1,cd1] = dwt(s,'haar');
ss = idwt(ca1,cd1,'db2'); %一维离散单尺度小波变换系数的重建
subplot(211); plot(s); title('Original signal'); %绘制原始信号
subplot(212); plot(ss); title('Reconstructed signals'); %绘制重构后的信号

```

执行上述程序，将生成如图 18.6 所示的一维离散单尺度小波变换系数的重建信号。

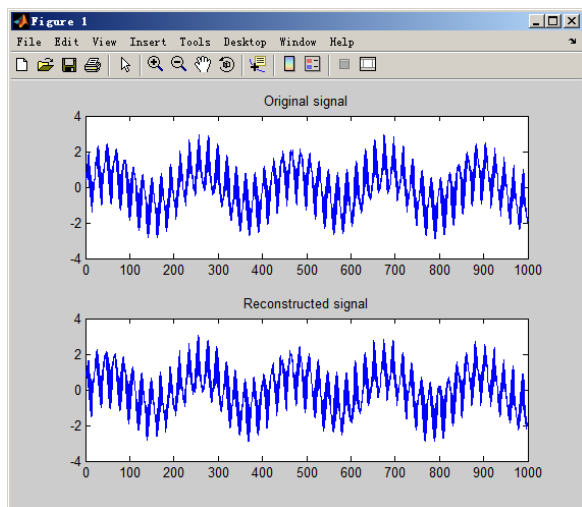


图 18.6 一维离散单尺度小波变换系数的重建

函数 `waverec()` 可用于一维离散多尺度小波变换系数的重建，其调用格式如下。

- `X = waverec(C,L,'wname')`: 输入参数 `C` 和 `L` 为一维离散多尺度小波变换的分解结构，`wname` 为小波函数，返回重构的信号 `X`。
- `X = waverec(C,L,Lo_R,Hi_R)`: 使用低通滤波器 `Lo_R` 和高通滤波器 `Hi_R` 重构信号 `X`。

【例 18.7】 一维离散多尺度小波变换系数的重建。

```
>> load leleccum; s = leleccum(1:500);
[C,1] = wavedec(s,3,'db5');
ss = waverec(c,1,'db5'); %一维离散多尺度小波变换系数的重建
err = norm(s-ss) %重构误差计算
subplot(211); plot(s); title('Original signal'); %绘制原始信号
subplot(212); plot(ss); title('Reconstructed signal'); %绘制重构信号
err =
4.275599904729175e-010
```

执行上述程序，将生成如图 18.7 所示的一维离散多尺度小波变换系数的重建信号。

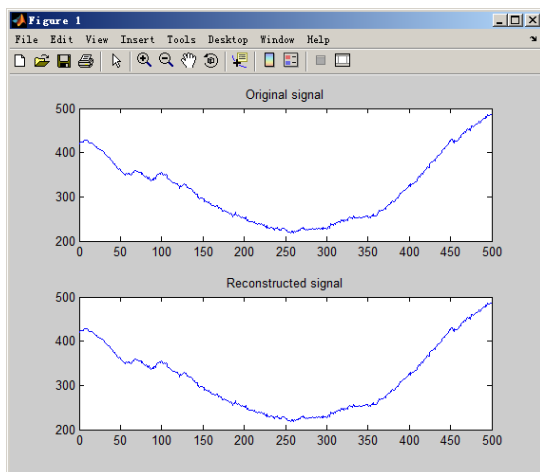


图 18.7 一维离散多尺度小波变换系数的重建

18.2.2 二维小波分析

MATLAB 小波分析工具箱还提供了二维小波分析，使我们可以方便地处理图像等二维数据。

类似于一维小波分析，可以进行的二维小波分析包括单尺度的离散小波变换、多尺度的离散小波变换和小波系数的提取，下面逐一介绍这些功能实现的函数。

1. dwt2()函数

函数 `dwt2()` 可用于二维单尺度的离散小波变换，其调用格式如下。

- `[cA,cH,cV,cD] = dwt2(X,'wname')`: 计算二维矩阵 `X` 离散小波变换的近似系数 `cA` 和细节系数 `cH` (水平)、`cV` (垂直)、`cD` (对角)，参数 `wname` 用于指定小波函数。
- `[cA,cH,cV,cD] = dwt2(X,Lo_D,Hi_D)`: 利用低通滤波器 `Lo_D` 和高通滤波器 `Hi_D` 实现二维离散小波分析。

【例 18.8】 二维单尺度的离散小波变换。

```
load flujet;
[cA1,cH1,cV1,cD1] = dwt2(X,'db1');           %二维单尺度的离散小波变换
subplot(2,2,1);plot(cA1);title('Approximation coefficients ');%小波近似系数绘制
                                                    %水平方向的小波细节系数绘制
subplot(2,2,2);plot(cH1);title('Details coefficients for horizontal');
                                                    %垂直方向的小波细节系数绘制
subplot(2,2,3);plot(cV1);title('Details coefficients for vertical');
                                                    %对角方向的小波细节系数绘制
subplot(2,2,4);plot(cD1);title('Details coefficients for diagonal');
执行上述程序，将生成如图 18.8 所示的二维单尺度的离散小波变换。
```

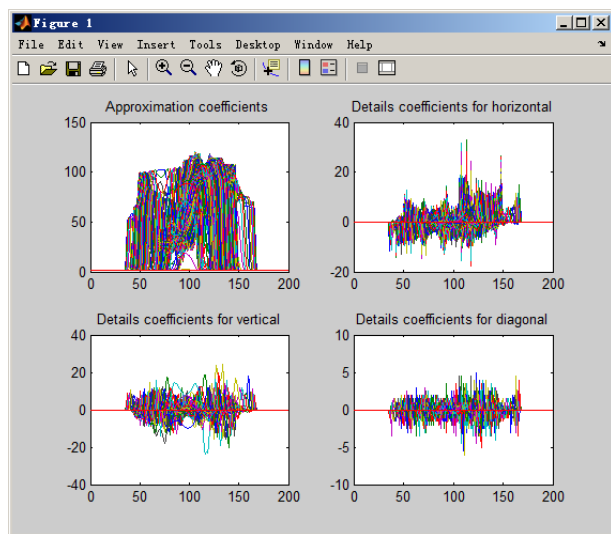


图 18.8 二维单尺度的离散小波变换

2. wavedec2()函数

函数 `wavedec2()` 可用于二维多尺度的离散小波变换，其调用格式如下。

- `[C,S] = wavedec2(X,N,'wname')`: 参数 `X` 为输入的二维矩阵信号，`N` 为小波尺度，`wname` 为小波函数，返回小波的分解信号 `C` 和 `S`。
- `[C,S] = wavedec2(X,N,Lo_D,Hi_D)`: 通过滤波器获取小波分解的信号 `C` 和 `S`。

【例 18.9】 二维多尺度的离散小波变换。

```
>> load flujet;
[c,s] = wavedec2(X,2,'db1');           %二维多尺度的离散小波变换
size(c)
ans =
     1    120000
>> disp(s)
```

```

100    75
100    75
200   150
400   300

```

3. detcoef2()函数

函数 `detcoef2()` 可用于提取高频的二维离散变换的细节小波系数，其调用格式如下。

`D = detcoef2(O,C,S,N)`: 函数的输入参数来自二维离散多尺度小波变换函数 `wavedec2()` 的分解信号 `[C,S]`，`N` 为指定的尺度，返回二维细节小波系数。

【例 18.10】 提取高频二维离散变换的细节小波系数。

```

load flujet;
[c,s] = wavedec2(X,2,'db1');
[chd2,cvd2,cdd2] = detcoef2('all',c,s,2);           %提取高频二维离散变换的细节小波系数
subplot(2,2,1);imshow(X,map);title('Original signal ');%原始图像绘制
                                                    %水平方向的细节信号绘制
subplot(2,2,2);imshow(chd2,map);title('Details coefficients for horizontal');
                                                    %垂直方向的细节信号绘制
subplot(2,2,3);imshow(cvd2,map);title('Details coefficients for vertical');
                                                    %对角方向的细节信号绘制
subplot(2,2,4);imshow(cdd2,map);title('Details coefficients for diagonal');
执行上述程序，将生成如图 18.9 所示的高频二维离散变换的细节小波系数的提取。

```

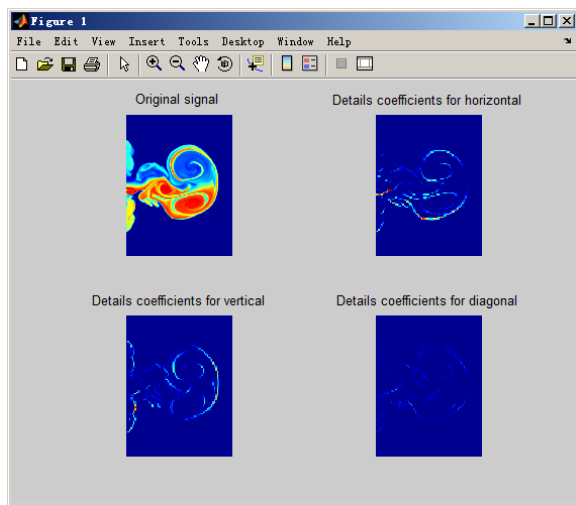


图 18.9 高频二维离散变换的细节小波系数的提取

4. appcoef2()函数

函数 `appcoef2()` 可用于提取二维小波分析的近似系数，其调用格式如下。

- `A = appcoef2(C,S,'wname',N)`: 在尺度 `N` 上计算二维小波分析的近似系数，输入参数 `C` 和 `S` 为二维离散多尺度小波变换函数 `wavedec2()` 的分解信号 `[C,S]`，`wname` 为小波函数。
- `A = appcoef2(C,S,'wname')`: 在尺度 `size(S,1)-2` 上计算二维小波分析的近似系数。
- `A = appcoef2(C,S,Lo_R,Hi_R)`: 利用滤波器提取二维小波分析的近似系数。
- `A = appcoef2(C,S,Lo_R,Hi_R,N)`: 利用滤波器提取尺度 `N` 上的二维小波分析的近似系数。

【例 18.11】 提取二维离散变换的近似小波系数

```

load flujet;           %导入信号
[c,s] = wavedec2(X,2,'db1');
ca2 = appcoef2(c,s,'db1',2); %提取二维小波分析的近似系数
imshow(ca2,map)        %显示二维小波系数

```

执行上述程序，将生成如图 18.10 所示的二维小波分析的近似系数的提取。

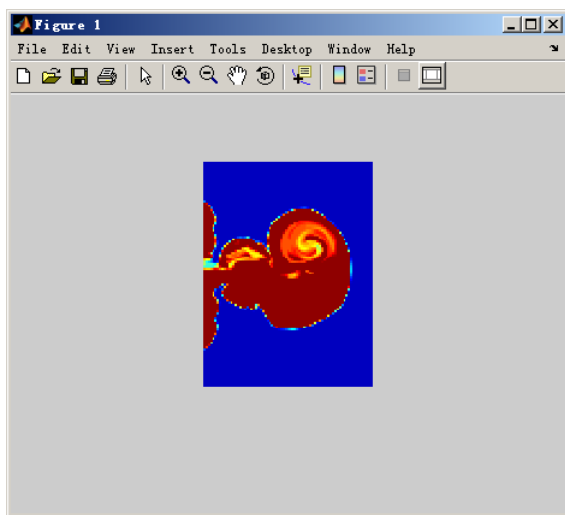


图 18.10 二维小波分析的近似系数的提取

18.2.3 小波包变换

小波包变换能够克服传统小波分析在高频区域频率分辨率差，在低频区域时间分辨率差的缺点，是一种更为有效的信号分析处理方法。

1. 一维小波包变换

在 MATLAB 小波分析工具中函数 `wpdec()` 可用于一维小波包变换，其调用格式如下。

- `T = wpdec(X,N,'wname',E,P)`: 参数 `X` 为一维信号，`N` 为小波包分解的尺度，`wname` 为使用的小波包的名称，参数 `E` 为小波变换时使用的熵的类型，`P` 为熵的相关参数，返回小波树 `T`。
- `T = wpdec(X,N,'wname')`: 等价于 `T = wpdec(X,N,'wname','shannon')`。
- 函数 `wpccoef()` 用于提取一维小波包变换的系数，其调用格式如下。
- `X = wpccoef(T,N)`: 参数 `T` 为函数 `wpdec()` 返回的小波树，`N` 为小波包的节点数，返回一维小波包分解的系数 `X`。
- `X = wpccoef(T)`: 等价于 `X = wpccoef(T,0)`。

【例 18.12】 一维小波包变换及其系数的提取。

```
load leleccum; s = leleccum(1:500);
figure(1); subplot(211);
plot(s); title('Original signal');           %绘制原始信号
wpt = wpdec(s,3,'db1')                       %一维小波包变换
Wavelet Packet Object Structure
=====
Size of initial data      : [1 500]
Order                    : 2
Depth                   : 3
Terminal nodes           : [7 8 9 10 11 12 13 14]
-----
Wavelet Name              : db1
Low Decomposition filter  : [ 0.7071    0.7071]
High Decomposition filter : [-0.7071    0.7071]
Low Reconstruction filter : [ 0.7071    0.7071]
High Reconstruction filter: [ 0.7071   -0.7071]
```

```

-----
Entropy Name           : shannon
Entropy Parameter      : 0
-----

```

```

plot(wpt);
cfs = wpccoef(wpt,[2 1]);           %提取小波包变换的系数
figure(1);subplot(212);
plot(cfs); title('Packet (2,1) coefficients');

```

执行上述程序，将生成如图 18.11 所示的一维小波包变换的系数和图 18.12 所示的一维小波包变换的树形结构图。

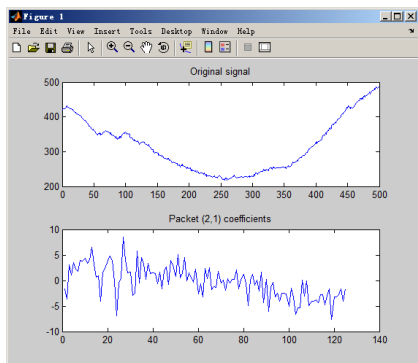


图 18.11 一维小波包变换的系数

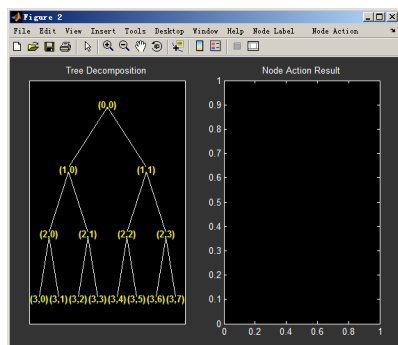


图 18.12 一维小波包变换的树形结构

函数 `wprec()` 可用于一维小波包分解的重构，其调用格式如下。

`X = wprec(T)`: 参数 `T` 为小波包分解函数 `wpdec()` 返回的小波树，`X` 为重构的向量。

函数 `wprcoef()` 可用于一维小波包某一节点处信号的重构，其调用格式如下。

`X = wprcoef(T,N)`: 重构节点 `N` 处的小波系数。

【例 18.13】 一维小波包变换的重构。

```

load leleccum; s = leleccum(1:500);
T= wpdec(s,3,'db1');           %小波包变换
X1 = wprec(T);                 %小波包变换的重构
subplot(211); plot(X1);title('一维小波包重构信号');
X2 = wprcoef(T,[2,1]);
subplot(212); plot(X2);title('一维小波包节点 (2, 1) 处的重构信号');

```

执行上述程序，将生成如图 18.13 所示的一维小波包变换的重构信号。

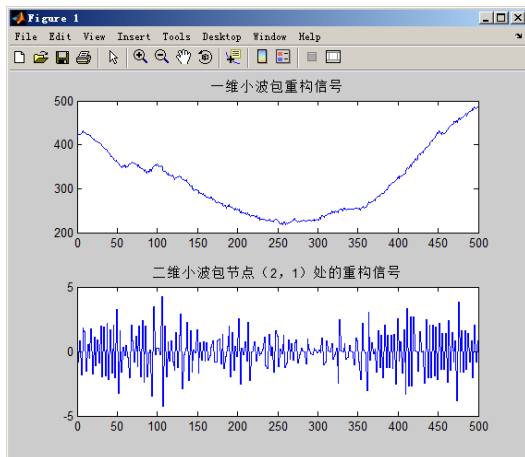


图 18.13 一维小波包变换的重构信号

2. 二维小波包变换

MATLAB 中可实现二维小波包变换, 包括小波包分解、系数提取、信号重构。其中: 函数 `wpdec2()` 可用于二维小波包变换的分解, 其调用格式如下。

- `T = wpdec2(X,N,'wname',E,P)`: `X` 为二维的输入信号, `N` 为小波包分解的尺度, `E` 为小波变换时使用的熵的类型, `P` 为熵的相关参数, 返回小波树 `T`。

- `T = wpdec2(X,N,'wname')`: 等价于 `T = wpdec2(X,N,'wname','shannon')`。

二维小波包系数的提取同一维小波包系数的提取, 也可以使用函数 `wpcoef()`。

函数 `wprec2()` 用于二维小波包信号重构, 其调用格式如下。

`X = wprec2(T)`: 参数 `T` 为二维小波包, 返回重构的信号 `X`。

【例 18.14】 二维小波包变换及其系数的提取。

```
>> load flujet;
T = wpdec2(X,3,'db2') %二维小波包变换
Wavelet Packet Object Structure
=====
Size of initial data      : [400 300]
Order                    : 4
Depth                    : 3
Terminal nodes           : [21 22 23 24 25 26 27 28 29 30 31 32 33 34 35 36 ...]
-----
Wavelet Name              : db2
Low Decomposition filter  : [-0.1294    0.2241    0.8365    0.483]
High Decomposition filter : [ -0.483    0.8365   -0.2241   -0.1294]
Low Reconstruction filter : [ 0.483    0.8365    0.2241   -0.1294]
High Reconstruction filter: [-0.1294   -0.2241    0.8365   -0.483]
-----
Entropy Name              : shannon
Entropy Parameter         : 0
-----

plot(T);
figure;
subplot(2,2,1);imshow(X,map);title('原始信号');
X1 = wpcoef(T,[2,1]);
subplot(2,2,2);imshow(X1,map);title('节点(2,1)处的小波系数');
X2 = wpcoef(T,[2,2]);
subplot(2,2,3);imshow(X2,map);title('节点(2,2)处的小波系数');
X3 = wprec2(T);
subplot(2,2,4);imshow(X3,map);title('重构的小波信号');
```

执行上述程序将生成如图 18.14 所示的二维小波包变换的树形结构和图 18.15 所示的二维小波包变换的系数提取和重构图。

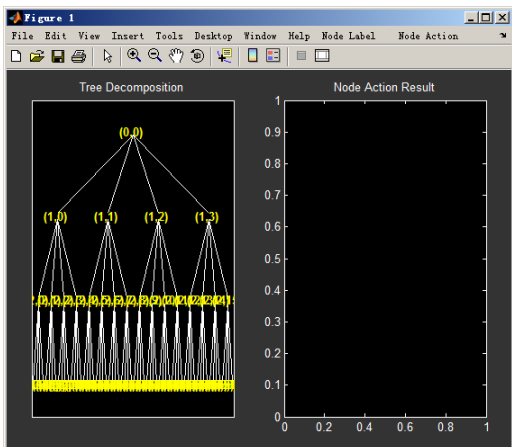


图 18.14 二维小波包变换的树形结构

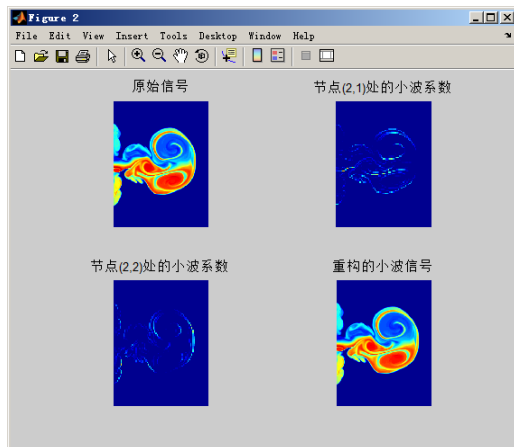


图 18.15 二维小波包变换的系数提取和重构

18.2.4 信号去噪

小波分析在信号去噪方面的应用也越来越广泛。基于小波分析的阈值去噪是比较常用的信号去噪方法,该方法的基本思想是对小波分解后的系数中大于或小于某阈值的系数进行一定的处理,处理后的小波系数重构为去噪后的信号。

函数 `ddencmp()`、`thselect()`、`wbmpen()`和 `wdcbm()`可用于获取信号去噪的阈值。下面具体介绍这几个函数的使用方法。

函数 `ddencmp()`可用于信号去噪或压缩的默认阈值的获取,其调用格式如下。

- `[THR,SORH,KEEPAPP,CRIT] = ddencmp(IN1,IN2,X)`: 输入参数 `IN1` 可为 `den` (去噪)或 `cmp` (压缩), `IN2` 可为 `wv` (小波分析)或 `wp` (小波包分析), 参数 `X` 为一维或者二维的输入信号, 函数返回的 `THR` 为阈值, `SORH` 为软阈值或硬阈值的选择参数, `KEEPAPP` 为保存的近似系数, `CRIT` 为熵名, 只在小波包分析的时候返回。
- `[THR,SORH,KEEPAPP] = ddencmp(IN1,'wv',X)`: 该调用格式使用小波分析对信号处理。
- `[THR,SORH,KEEPAPP,CRIT] = ddencmp(IN1,'wp',X)`: 该调用格式使用小波包分析对信号处理。

函数 `thselect()`可用于信号去噪阈值的选择,其调用格式如下。

`THR = thselect(X,TPTR)`: 参数 `TPTR` 用于定义阈值选择的规则,包括 `rigrsure` (自适应阈值选择)、`heursure` (启发式阈值选择)、`sqtwolog` (阈值为 $\sqrt{2 \cdot \log(\text{length}(X))}$) 和 `minimaxi` (极大极小值阈值选择), 返回选择的信号 `X` 的阈值 `THR`。

函数 `wbmpen()`可用于一维和二维信号去噪阈值的选择,其调用格式如下。

`THR = wbmpen(C,L,SIGMA,ALPHA)`: 参数 `C` 和 `L` 为去噪信号的小波分解结构, 参数 `SIGMA` 为零均值高斯白噪声降噪模型的标准差, `ALPHA` 为惩罚的阈值选择算法 Birge–Massart 的参数, 一般应大于 1, 返回选择的阈值。

函数 `wdcbm()`通过 Birge–Massart 策略选择小波消噪的阈值,其调用格式如下。

- `[THR,NKEEP] = wdcbm(C,L,ALPHA)`: 参数 `C` 和 `L` 为尺度 $\text{length}(L)-2$ 上的去噪信号的小波分解结构, 算法参数 `ALPHA` 需要为大于 1 的实数, 返回选择的阈值 `THR` 和系数的个数 `NKEEP`。
- `[THR,NKEEP] = wdcbm(C,L,ALPHA,M)`: 参数 `M` 需要为大于 1 的实数。

函数 `wden()`、`wdencomp()`和 `wpdencmp()`可用于信号的阈值去噪。下面具体介绍这几个函数的使用方法。

函数 `wden()`可用于一维信号的自动消噪,其调用格式如下。

- `[XD,CXD,LXD] = wden(X,TPTR,SORH,SCAL,N,'wname')`: 参数 `X` 为需要消噪的一维信号, `TPTR` 为阈值选择规则, 包括 `rigrsure` (Stein 无偏似然估计准则)、`heursure` (启发式阈值选择)、`sqtwolog` (采用阈值 $\sqrt{2 \log(\cdot)}$) 和 `minimaxi` (极大极小值阈值选择), 参数 `SORH` 的值可为 `s` (软阈值)或 `h` (硬阈值), 参数 `SCAL` 用于控制阈值需要重新调整, 包括 `one` (不调整)、`sln` (根据第一层的系数估计的噪声水平调整阈值)和 `mln` (根据不同层次的噪声水平调整阈值), 参数 `N` 为小波分解的层数, 参数 `wname` 为使用的小波函数名称, 函数返回消噪后的信号 `XD` 和消噪后信号的小波分解结构 `CXD` 和 `LXD`。
- `[XD,CXD,LXD] = wden(C,L,TPTR,SORH,SCAL,N,'wname')`: 参数 `C` 和 `L` 为去噪信号的小波分解结构。

函数 `wdencomp()`可用于一维或二维信号的降噪或压缩,其调用格式如下。

- `[XC,CXC,LXC,PERF0,PERFL2] = wdencmp('gbl',X,'wname',N,THR,SORH,KEEPAPP)`: 输入参数 `X` 为一维或二维信号, 参数 `gbl` 表示每层都使用相同的阈值进行处理, `N` 为小波压缩的尺度, `wname` 为小波函数名称, `THR` 为阈值向量, `SORH` 为软阈值或硬阈值, `KEEPAPP` 为 1 细节系数不能阈值化, 返回的参数包括消噪或压缩后的信号 `XC`、`CXC` 和 `LXC` 小波分解的结构, `PERF0` 和 `PERFL2` 为恢复和压缩的范数百分比。
- `[XC,CXC,LXC,PERF0,PERFL2] = wdencmp('lvd',X,'wname',N,THR,SORH)`: 参数 `lvd` 表示每层使用不同的阈值进行处理。
- `[XC,CXC,LXC,PERF0,PERFL2] = wdencmp('lvd',C,L,'wname',N,THR,SORH)`: 参数 `C` 和 `L` 为去噪信号的小波分解结构。

函数 `wpdencmp()` 可用于使用小波包降噪或压缩信号, 其调用格式如下。

- `[XD,TREED,PERF0,PERFL2] = wpdencmp(X,SORH,N,'wname',CRIT,PAR,KEEPAPP)`: 参数 `X` 为需要消噪或压缩的一维或二维信号, `SORH` 为软阈值或硬阈值, `N` 为小波分解的层数, `wname` 为小波函数的名称, `CRIT` 为小波包的熵标准, `PAR` 为阈值参数, `KEEPAPP` 为 1 细节系数不能阈值化, 函数返回消噪或压缩后的信号 `XD`, 小波分解树 `TREED`、`PERF0` 和 `PERFL2` 为恢复和压缩的能量百分比。
- `[XD,TREED,PERF0,PERFL2] = wpdencmp(TREE,SORH,CRIT,PAR,KEEPAPP)`: 输入小波树进行信号的消噪和压缩。

【例 18.15】 利用函数 `ddencmp()` 获取小波消噪的阈值, 并分别使用小波分析和小波包分析对一维信号消噪。

```
>> load leleccum;
s = leleccum(1:500);
X=s+10*randn(1,500);
[THR,SORH,KEEPAPP] = ddencmp('den','wv',X)           %利用函数 ddencmp() 获取小波消噪的阈值
THR =
    33.18040314788969
SORH =
s
KEEPAPP =
    1
>> XC = wdencmp('gbl',X,'sym4',2,THR,SORH,KEEPAPP); %使用小波分析对信号消噪
[THR,SORH,KEEPAPP,CRIT] = ddencmp('den','wp',X)      %利用函数 ddencmp() 获取小波包消噪的阈值
XD= wpdencmp(X,SORH,3,'db2',CRIT,THR,KEEPAPP);      %使用小波包分析对信号消噪
THR =
    4.10073741773467
SORH =
h
KEEPAPP =
    1
CRIT =
sure
subplot(221); plot(s); title('原始信号');
subplot(222); plot(X); title('加噪后的信号');
subplot(223); plot(XC); title('小波分析去噪后的信号');
subplot(224); plot(XD); title('小波包去噪后的信号');
```

执行上述程序, 将生成如图 18.16 所示的小波阈值消噪效果图。

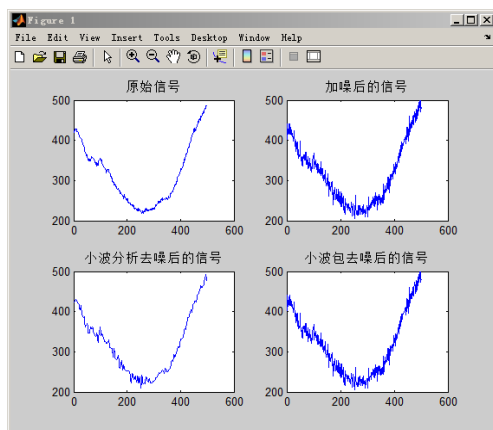


图 18.16 小波阈值一维信号降噪效果图

【例 18.16】利用函数 `wbmpen()` 获取小波降噪的阈值，并分别使用小波分析对二维信号降噪。

```
load flujet;
subplot(2,2,1);imshow(X,map);title('原始图像');
X=X+10*randn(size(X)); %信号加噪
subplot(2,2,2);imshow(X,map);title('加噪后的图像');
[c,s] = wavedec2(X,3,'db3');
det1 = detcoef2('compact',c,s,1);
sigma = median(abs(det1))/0.6745;
alpha = 1.2;
thr = wbmpen(c,s,sigma,alpha) %利用函数 wbmpen() 获取小波降噪的阈值 thr
xd = wdenncmp('gbl',c,s,'db3',3,thr,'s',1); %小波去噪
subplot(2,2,3);imshow(xd,map);title('去噪后的图像');
```

执行上述程序将生成如图 18.17 所示的小波阈值二维信号降噪效果图。

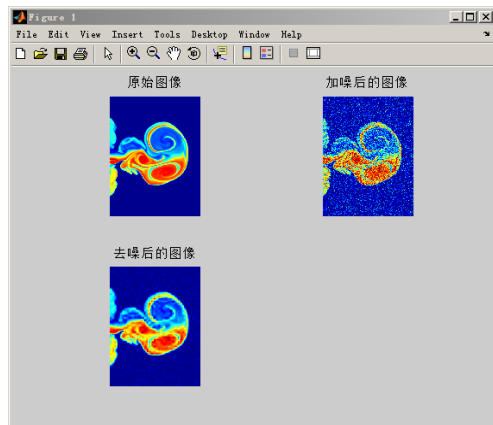


图 18.17 小波阈值二维信号降噪效果图

【例 18.17】利用函数对信号进行自动降噪。

```
load leleccum; s = leleccum(1:500);
X=s+5*randn(1,500);
[XD,CXD,LXD] = wden(X,'rigrsure','s','one',3,'db3') %利用函数 wden() 对信号进行自动降噪
subplot(2,2,1);plot(s);title('原始信号')
subplot(2,2,2);plot(X);title('带噪声的信号')
subplot(2,2,3);plot(XD);title('消噪后的信号')
subplot(2,2,4);plot(CXD);title('消噪后的小波分解系数')
```

执行上述程序，将生成如图 18.18 所示的自动降噪效果图

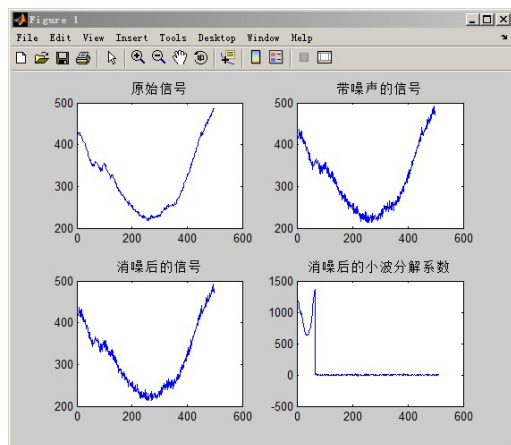


图 18.18 小波阈值二维信号消噪效果图

18.2.5 信号压缩

函数 `ddencmp()` 和 `wdcbm()` 可用于信号压缩阈值的获取，而函数 `wdencmp()` 和 `wpdencmp()` 可用于信号压缩，这些函数的具体使用在前面已经详细叙述过，这里以简单的实例演示这些函数的使用。

【例 18.18】 利用函数对信号压缩。

```
load leleccum; s = leleccum(1:500);
[c,l]=wavedec(s,3,'sym3')
[THR,SORH,KEEPAPP] = ddencmp('cmp','wv',s); %获取信号压缩阈值
XC = wdencmp('gbl',c,l,'sym3',3,THR,SORH,KEEPAPP); %信号压缩
subplot(2,1,1);plot(s);title('原始信号');
subplot(2,1,2);plot(XC);title('压缩信号');
```

执行上述程序，将生成如图 18.19 所示的小波信号压缩效果图。

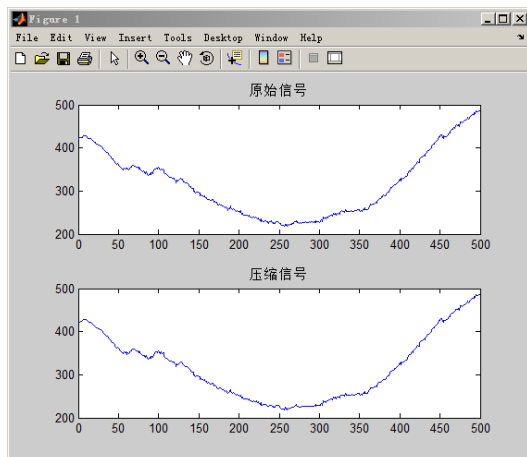


图 18.19 小波信号压缩效果图

18.3 利用 GUI 实现小波分析

前面介绍了利用函数法实现小波分析，但对于不擅长编程的用户可能使用还是不方便，本节

主要介绍如何利用图形用户界面实现小波分析。

18.3.1 小波分析工具箱 GUI 的启动

小波分析工具箱 GUI 的启动方式有如下两种。

- 启动 MATLAB 在命令窗口输入命令 `wavemenu`。
- 选择 “Start” → “Toolboxes” → “More” → “Wavelet” → “Wavelet Toolbox Main Menu” 命令。

执行上述操作后将打开如图 18.20 所示的小波分析工具箱 GUI。

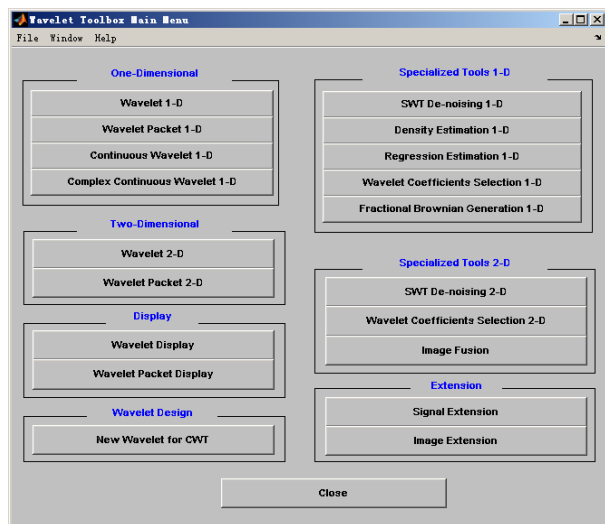


图 18.20 小波分析工具箱 GUI

18.3.2 小波分析工具箱 GUI 的工作界面

小波分析工具箱 GUI 的主界面主要包括一维小波分析工具、二维小波分析工具、小波显示工具、小波函数设计工具、一维小波分析的专用工具、二维小波分析的专用工具、扩展小波工具。

1. 一维小波分析工具

一维小波分析工具主要包括 Wavelet 1-D（一维小波变换）、Wavelet Packet 1-D（一维小波包变换）、Continuous Wavelet 1-D（连续的一维小波变换）和 Complex Continuous Wavelet 1-D（复杂的连续一维小波变换）。

2. 二维小波分析工具

二维小波分析工具主要包括 Wavelet 2-D（二维小波变换）和 Wavelet Packet 2-D（二维小波包变换）。

3. 小波显示工具

小波显示工具包括 Wavelet Display（小波显示）和 Wavelet Packet Display（小波包显示）。

4. 小波函数设计工具

小波函数设计工具 New wavelet for cwt（设计新的小波用于连续小波变换）。

5. 一维小波分析的专用工具

一维小波分析的专用工具包括 SWT De-noising 1-D（一维平滑小波降噪）、Density

Estimation 1-D(一维密度估计小波变换)、Regression Estimation 1-D(一维回归估计小波变换)、Wavelet Coefficients Selection 1-D(一维小波系数选择)和 Fractional Brownian Generation 1-D(一维小波分数布朗生成)。

6. 二维小波分析的专用工具

二维小波分析的专用工具包括 SWT De-noising 2-D(二维平滑小波降噪)、Wavelet Coefficients Selection 2-D(一维小波系数选择)和 Image Fusion(影像融合)

7. 小波扩展工具

小波扩展工具包括 Signal Extension(信号扩展)和 Image Extension(影像扩展)。

18.3.3 小波分析工具箱的操作

本节小波分析工具箱操作的讲解以最常用的一维和二维小波变换为例,向广大用户演示如何利用小波工具箱的 GUI 实现一维和二维小波变换。

【例 18.19】利用小波分析工具箱 GUI 实现一维小波变换。

(1) 在命令窗口输入 wavemenu, 启动 MATLAB 小波分析工具箱 GUI, 单击一维小波分析的“Wavelet 1-D”按钮, 将弹出如图 18.21 所示的一维小波分析 GUI 窗口。

一维小波分析 GUI 窗口包括 6 个菜单项, 其中,

- “File” 菜单项: 主要用于小波分析 GUI 数据的导入和导出。
- “View” 菜单项: 主要用于界面工具栏等的显示控制。
- “Insert” 菜单项: 可用于在绘制的图形中添加坐标轴、图形等。
- “Tools” 菜单项: 提供了一些图形处理的工具。
- “Window” 菜单项: 用于控制窗口的显示。
- “Help” 菜单项: 方便用户获取小波分析的帮助文档。

在界面的主区域将显示小波变换后的结果, 其右侧区域包括小波变换参数的设置。

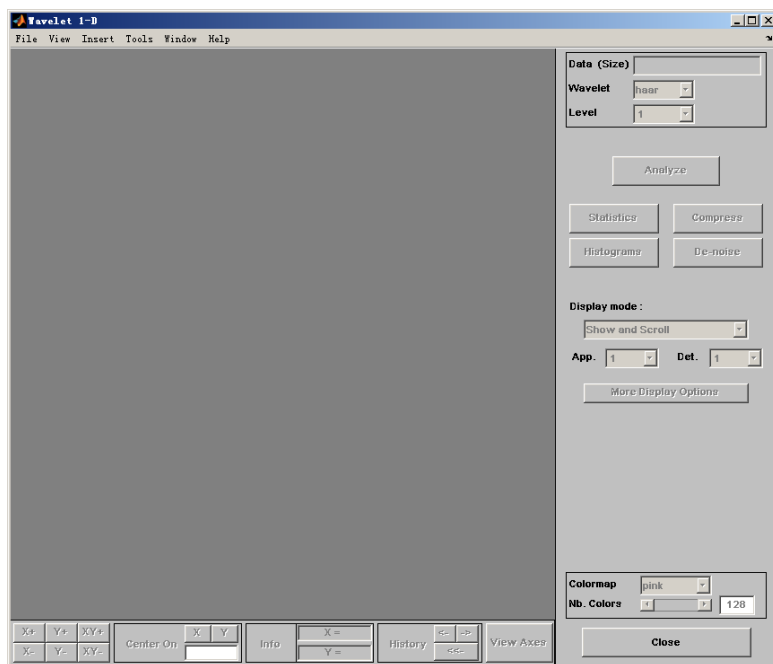


图 18.21 一维小波分析 GUI 窗口

(2) 在一维小波分析 GUI 窗口中通过选择“File”→“Load”→“Singal”命令导入信号,在本章程序路径下导入数据 data,一维小波分析 GUI 窗口将显示信号的波形。

在窗口的右侧区域设置小波变换的 Wavelet (小波函数) 和 Level (分解的层数), 本例中我们设置小波函数为“Haar”, 小波分解的层数为 5。设置完成后单击“Analyze”按钮, 将启动小波分析, 在窗口中将显示小波分析后信号各层次的图, 如图 18.22 所示。

此时之前的许多灰色的按钮变为激活状态, 用户可以使用, 在“Display mode”下拉列表框中可选择小波信号分解的显示模式, 其中包括如下选项。

- Show and Scroll (滚动显示模式): 显示原始信号和近似信号的叠加、细节信号、小波细节系数, 同时在下方的 App 和 Det 下拉列表框中可以选择近似信号和细节信号的尺度。
- Full Decomposition (完全分解模式): 原始信号、所有的近似信号和细节信号将显示出来, 为 MATLAB 默认的显示方式, 图 18.22 中的显示模式即为该模式。
- Separate Mose (分离模式): 将原始信号, 各层的近似信号和细节信号分为两列显示。
- Superimpose Mode (叠加模式): 将近似信号和细节信号的各层系数叠加显示。
- Tree Mode (树模式): 显示小波分解的树形结构。
- Show and Scroll (Stem Cfs) (滚动柱状图显示模式): 小波的细节系数使用柱状图显示。

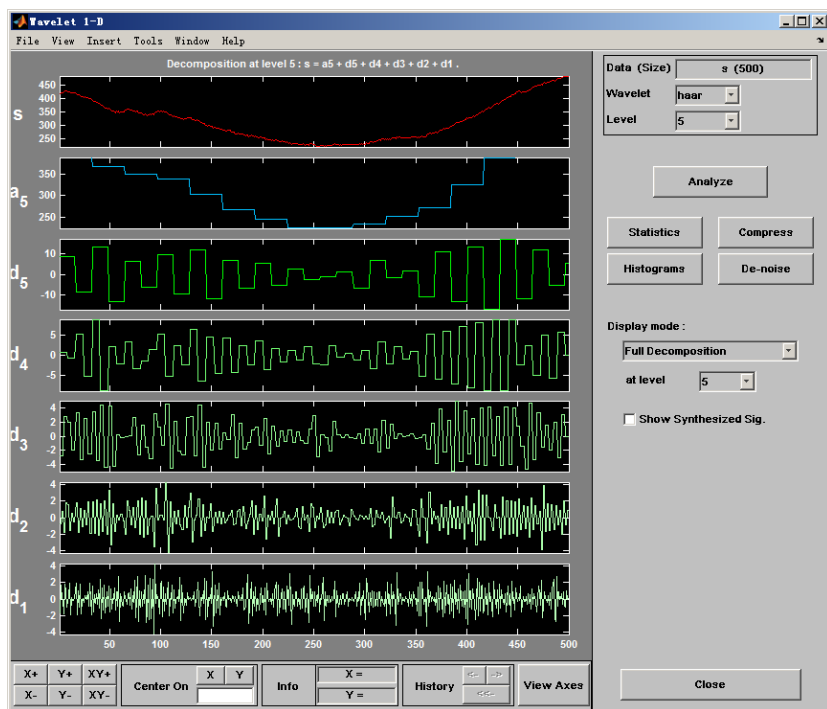


图 18.22 一维小波分析

(3) 在一维小波分析 GUI 窗口中“Analyze”按钮下方提供了 4 个按钮: Statistics (显示信号的统计特性)、Compress (信号压缩)、Histograms (显示信号的直方图) 和 De-noise (信号去噪)。下面依次讲解这 4 个按钮的功能。

单击一维小波分析 GUI 窗口中“Statistics”按钮, 将打开如图 18.23 所示的一维小波分析的统计图。可用于查看原始信号、细节信号、近似信号和重构信号的统计参数, 包括均值、最大值、最小值、标准差等常用的统计量和直方图、累计直方图等统计图。

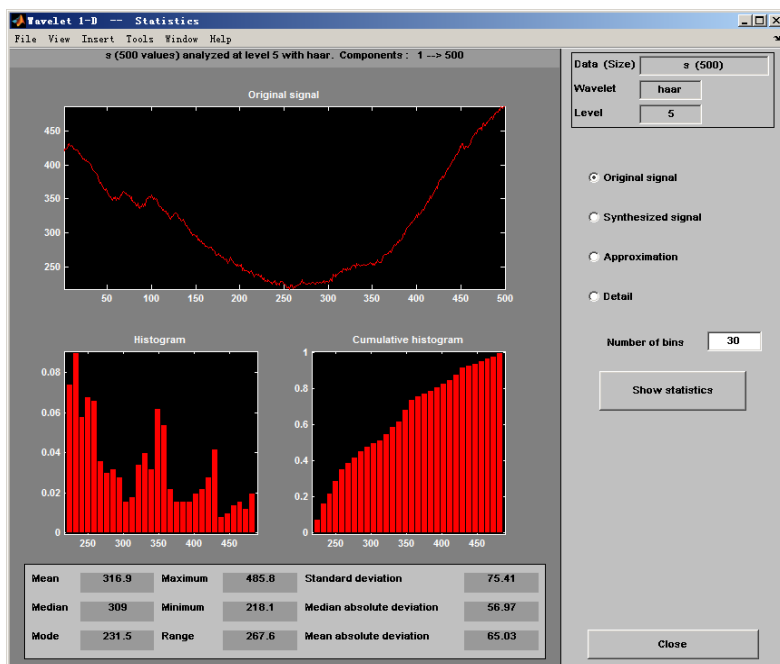


图 18.23 一维小波分析的统计图

单击一维小波分析 GUI 窗口中“Compress”按钮，将打开如图 18.24 所示的一维小波压缩的界面。可以使用全局阈值或局部阈值进行信号压缩。

单击一维小波分析 GUI 窗口中“Histograms”按钮，将打开如图 18.25 所示的一维小波分析直方图显示的界面。可用于查看原始信号、细节信号、近似信号和重构信号的直方图分布。

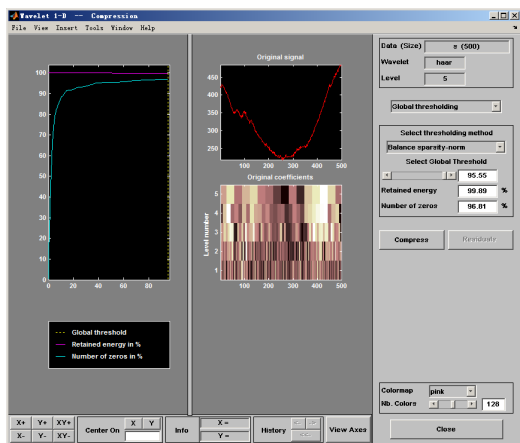


图 18.24 一维小波压缩窗口

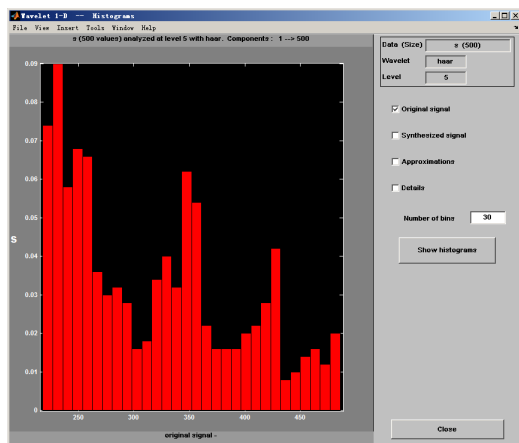


图 18.25 一维小波分析直方图

单击一维小波分析 GUI 窗口中“De-noise”按钮，将打开如图 18.26 所示的一维小波消噪的界面。可以采用阈值去噪法对信号去噪，需要设置的参数有 select thresholding method（选择阈值的方法）、select noise structure（选择噪声的结构）等。

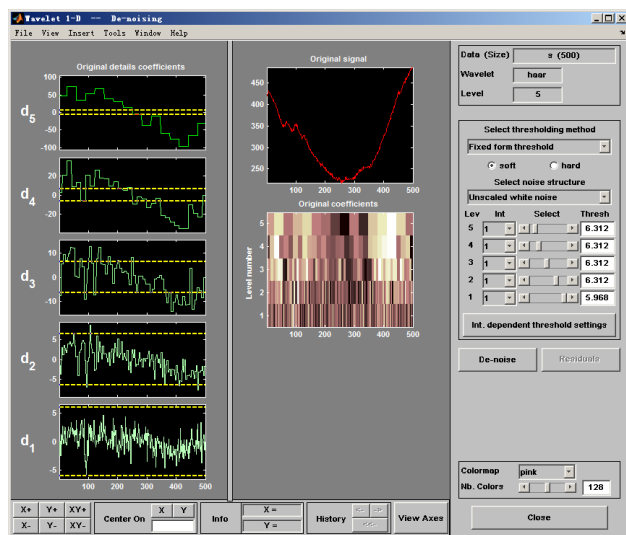


图 18.26 一维小波消噪的窗口

【例 18.20】 利用小波分析工具箱 GUI 实现二维小波变换。

(1) 在命令窗口输入 `wavemenu`，启动 MATLAB 小波分析工具箱 GUI，单击二维小波分析的“Wavelet 2-D”按钮，将弹出如图 18.27 所示的二维小波分析 GUI 窗口。二维小波分析 GUI 窗口与一维小波分析 GUI 窗口的界面布局类似，这里不再详细展开叙述。

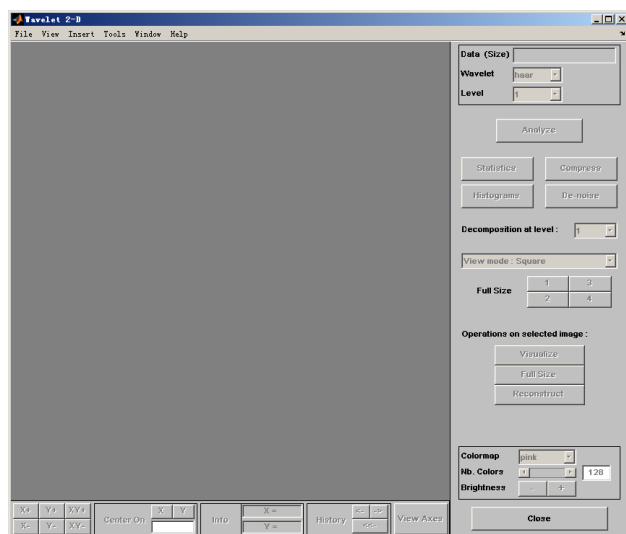


图 18.27 二维小波分析 GUI 窗口

(2) 在二维小波分析 GUI 窗口中通过选择“File”→“Load”→“Image”命令导入信号，在本程序路径下导入数据 `data1`，二维小波分析 GUI 窗口将显示图形信号。

在窗口的右侧区域设置小波变换的 Wavelet (小波函数) 和 Level (分解的层数)，本例中我们设置小波函数为“Haar”，小波分解的层数为 2。设置完成后单击“Analyze”按钮，将启动小波分析，在窗口中将显示图像的分解图，如图 18.28 所示。

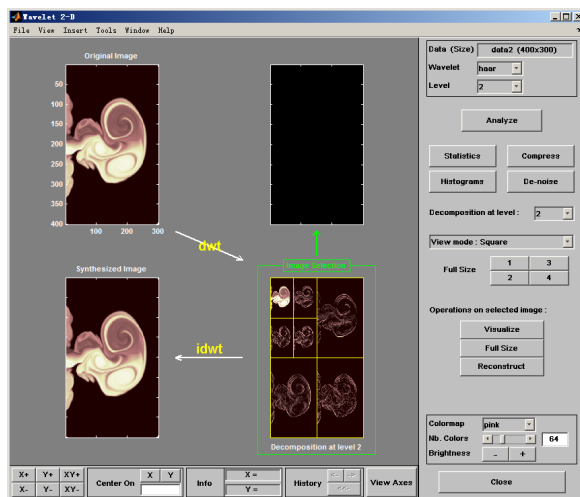


图 18.28 二维小波图像的分解图

在二维小波分析 GUI 窗口中可以选择的显示模式包括如下两种。

- View mode: Square (方块图模式显示): 图 18.28 即为该模式的显示, 在这种模式下将包括 4 个窗口, 左上方为原始图像, 右上方为用于供用户放大显示信号的窗口, 左下方为重构后的图像, 右下方为近似和细节信号。
- View mode: Tree (树形结构显示): 该模式将显示原始图像、重构的图像及不同层次的近似和细节信号, 如图 18.29 所示。

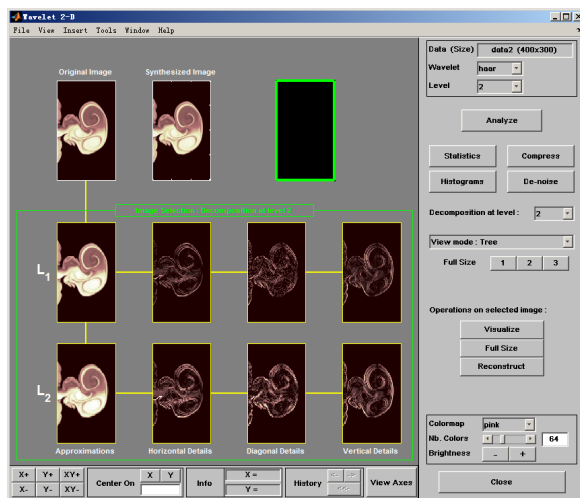


图 18.29 二维小波图像分解的树形结构图

(3) 在二维小波分析 GUI 窗口中的右下方有“Operations on selected image”按钮组, 包括“Visualize”、“Full Size”和“Reconstruct”按钮, 用于控制右下方选中的图像的显示模式。

- 单击“Visualize”按钮, 当显示模式为 View mode: Square 时, 在右上方将显示所选择的图像。
- 单击“Full Size”按钮, 将以最大化的方式显示图像。
- 单击“Reconstruct”按钮, 当显示模式为 View mode: Square 时, 在右上方将显示重构的图像。

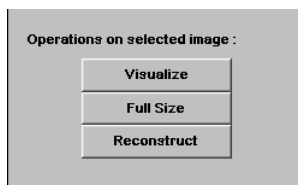


图 18.30 二维小波分析 GUI 窗口的“Operations on selected image”按钮组

(4) 在二维小波分析 GUI 窗口中“Analyze”按钮下方也提供了 4 个按钮：Statistics（显示信号的统计特性）、Compress（信号压缩）、Histograms（显示信号的直方图）和 De-noise（信号去噪）。下面依次讲解这 4 个按钮的功能。

单击二维小波分析 GUI 窗口中“Statistics”按钮，将打开如图 18.31 所示的二维小波分析的统计图。可用于查看原始信号、细节信号、近似信号和重构信号的统计参数，包括均值、最大值、最小值、标准差等常用的统计量和直方图、累计直方图等统计图。

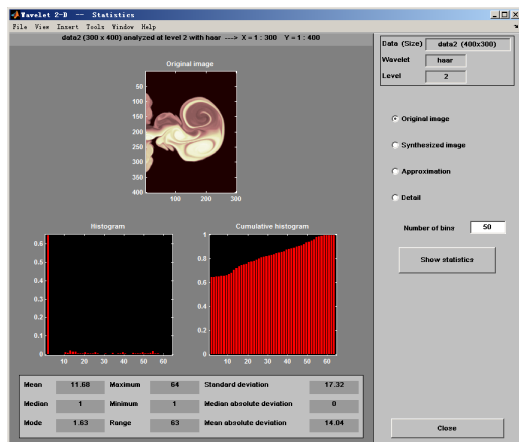


图 18.31 二维小波分析的统计图

单击二维小波分析 GUI 窗口中“Compress”按钮，将打开如图 18.32 所示的二维小波压缩的界面。可以使用全局阈值或局部阈值进行信号压缩。在设置完成选择阈值方法和相关参数后，单击“Compress”按钮将打开压缩后的图像，如图 18.33 所示。

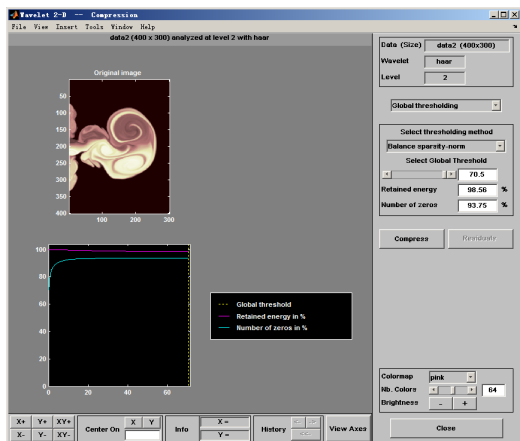


图 18.32 二维小波压缩窗口

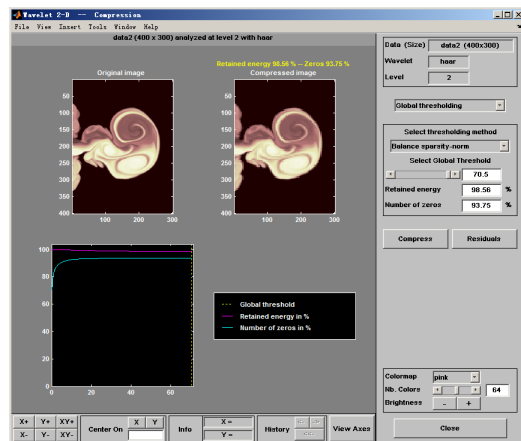


图 18.33 利用二维小波进行图像压缩

单击二维小波分析 GUI 窗口中“Histograms”按钮，将打开如图 18.34 所示的二维小波分析

直方图显示的界面。可用于查看原始信号、细节信号、近似信号和重构信号的直方图分布。

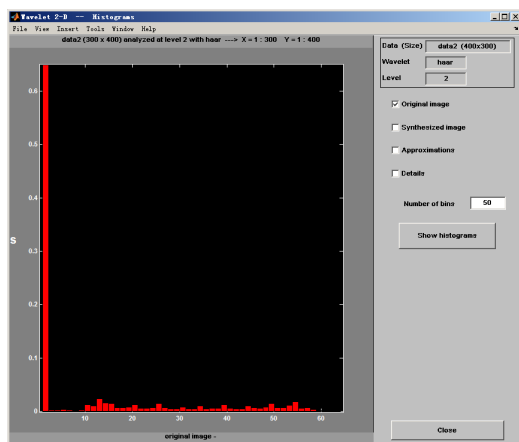


图 18.34 二维小波分析直方图

单击二维小波分析 GUI 窗口中“De-noise”按钮，将打开如图 18.35 所示的二维小波消噪的界面。可以采用阈值去噪法对信号去噪，需要设置的参数有 Select thresholding method (选择阈值的方法)、Select noise structure(选择噪声的结构)等。设置完成消噪的参数后，单击“De-noise”按钮，在图 18.36 中将显示消噪的图像。

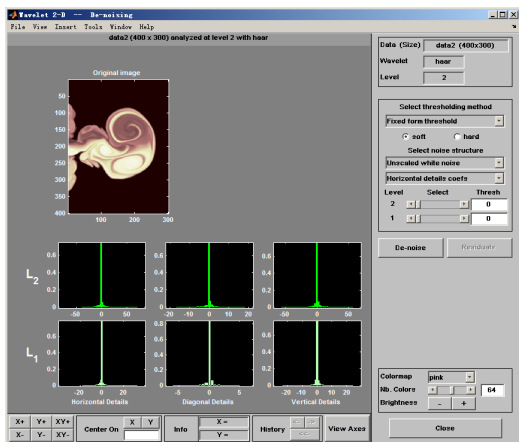


图 18.35 二维小波消噪的窗口

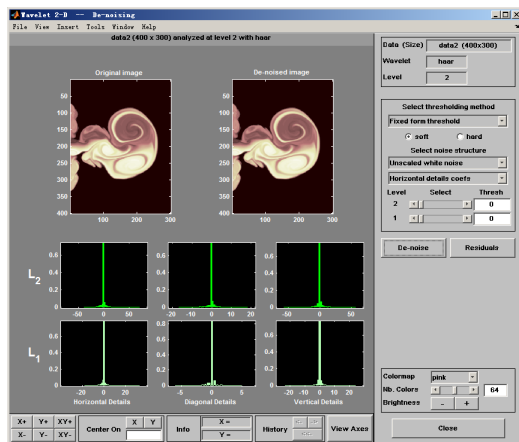


图 18.36 利用二维小波进行图像消噪

18.4 本章小结

本章主要介绍小波分析工具箱的使用，首先介绍了小波变换的基础知识，使读者对小波分析原理有基本的了解；然后介绍了常用的小波分析函数的使用，包括一维小波分析、二维小波分析、小波包变换、信号去噪、信号压缩；本章同时也介绍了如何利用 GUI 实现小波分析，其中包括了小波分析工具箱 GUI 的启动、工作界面的介绍，并以常用的一维和二维小波变换演示工具箱的使用。

小波分析近年来应用越来越广泛，相信通过本章的学习读者会对小波分析工具箱有初步的了解，其中还有很多功能限于篇幅等原因，笔者未详细介绍，读者可以阅读相关帮助文档，了解其使用方法。



第19章

遗传算法工具箱

遗传算法 (Genetic Algorithm) 是一类借鉴自然选择和遗传机制演化而来的随机化搜索方法。遗传算法在并行分布、系统建模、自适应控制、模式识别和图像处理等领域都有一定的应用。本章主要介绍 MATLAB 遗传算法与直接搜索工具箱的使用, 通过本章的学习读者将掌握如何利用软件实现遗传算法。

19.1 遗传算法的基本概念

遗传算法是一种新兴的技术, 它通过模拟生物进化的过程: 繁殖、交叉、变异和选择, 从初始种群开始发生遗传、变异, 以生物的进化准则“优胜劣汰, 适者生存”来实现随机优化与选择, 产生新的适应性更好的后代, 这样的过程不断反复迭代, 直至寻找到的最优解或满意解, 即对环境适应性最好的解。

基本的遗传算法主要包括以下步骤。

(1) 个体的编码。遗传算法在进行解的搜索之前, 需要先将解空间的数据表示成遗传算法空间的基因型串结构数据。

(2) 初始群体的产生。随机产生多个初始串结构数据, 每个串结构数据称为一个个体, 多个个体构成了一个群体, 遗传算法以这个群体作为初始群体, 开始后面的迭代算法。

(3) 适应度的计算。适应性用于表明个体或解的优劣性。适应度是个体对环境适应性好坏的度量, 是生物个体进化的重要参考变量。

(4) 选择计算。选择的目的是为了从群体中选出优良的个体, 繁殖子孙进入后面的生物进化过程。遗传算法通过选择过程体现这一思想, 对于适应度高的个体将有较大的机会被选择上, 有机会成为父代繁殖后代, 而适应度低的个体则很有可能被淘汰, 选择实现了达尔文的适者生存的原则。常用的选择算法有以下 3 种。

- ① 交叉计算。对于在选择操作中选中的个体, 对两个个体的相同位置的基因进行交换, 从而产生新的个体。这个过程具有一定的随机性, 可以产生新的基因组合。交叉计算可以分为单点交叉和多点交叉。
- ② 变异计算。对选中的个体将按一定的概率执行变异操作, 个体不同基因位上为 1 的变异为 0, 为 0 的变异为 1。
- ③ 迭代计算。重新步骤 3, 计算新的个体的适应度, 如果此时算法符合终止条件, 则停止算法, 返回最优解或满意解, 否则迭代重新计算步骤 4~6, 进行选择、交叉、变异操作, 直至满足算法的条件。

遗传算法具有以下 3 个特点。

1. 可行解的广泛性

遗传算法通过编码的方式构建群体, 搜寻可行解, 解的范围广泛, 同时在迭代的过程中, 随

着交叉、变异的进行，可以产生更多丰富的可行解，不易陷入局部最优解。

2. 算法的并行性

遗传算法作为一种新兴的搜索算法，与之前的搜索算法最大的不同在于传统的算法只能一个解的搜索，而遗传算法是以群体为单位进行搜索，算法具有较好的并行性。

3. 算法的智能性

遗传算法模拟生物进化过程具有较好的智能性，通过适应值大小对个体进行选择，使更适应环境变化的个体可以更好的繁殖后代。

19.2 遗传算法工具箱

在前面的小节中简单地介绍了遗传算法的基本原理和特点，读者可以发现遗传算法的功能是比较强大的，可以解决用户的许多优化、寻参、搜索等问题，而目前实现遗传算法除了用户完全自己手工编写代码外，比较常用且方便的工具就是 MATLAB。MATLAB 中的遗传算法与直接搜索工具箱为用户提供了函数法和 GUI 图形窗口两种方式实现遗传算法，本节将重点讲述如何在 MATLAB 中通过这两种方法实现遗传算法。

19.2.1 遗传算法相关函数

在 MATLAB 遗传算法与直接搜索工具箱中用于实现遗传算法的函数主要包括 `ga()`、`gaoptimget()`和 `gaoptimset()`，函数 `ga()`通过遗传算法搜寻函数的最小值，而函数 `gaoptimget()`和 `gaoptimset()`分别用于获取遗传算法的参数值和设置遗传算法的参数值。下面具体介绍这 3 个函数的用法。

1. `ga()`函数

函数 `ga()`为实现遗传算法的主要函数，可计算指定函数的最小值，其调用格式如下。

- `x = ga(fitnessfun, nvars)`: 其中 `fitnessfun` 为指定的计算最小值的函数，即适应度函数，参数 `nvars` 为优化函数的变量个数，即适应度函数输入参数的变量个数，返回最优解 `x`。
- `x = ga(fitnessfun, nvars, options)`: 参数 `options` 用于设置遗传算法，遗传算法的每个参数值存储在参数 `options` 的结构体中。
- `x = ga(problem)`: 该调用格式将上述调用格式中的参数 `fitnessfun`、`nvars` 和 `options` 存储在结构变量 `problem` 中。
- `[x, fval] = ga(...)`: 返回最优解 `x` 和最优解处的函数最小值 `fval`。
- `[x, fval, reason] = ga(...)`: 参数 `reason` 为包含算法终止原因的字符串。
- `[x, fval, reason, output] = ga(...)`: 参数 `output` 为结构体变量，用于描述遗传算法每一代的信息和算法的表现，包含 `randstate` (在遗传算法启动前随机产生的种群)、`randnstate` (在遗传算法启动前产生的正态随机种群)、`generations` (遗传算法计算的代数)、`funccount` (适应度函数估算的次数) 和 `message` (遗传算法终止的条件，同输出参数 `reason` 输出的内容)。
- `[x, fval, reason, output, population] = ga(...)`: 参数 `population` 返回最终的群体。
- `[x, fval, reason, output, population, scores] = ga(...)`: 参数 `scores` 返回最终群体的适应度值。

2. gaoptimset()函数

函数 gaoptimset()用于对遗传算法的参数进行设置，其调用格式如下。

- options = gaoptimset: 用于获取遗传算法默认参数的设置状况，具体如表 19.1 所示。
- gaoptimset: 返回遗传算法可以设置的所有参数及其可以设置的参数范围，具体如表 19.2 所示。
- options = gaoptimset('param1',value1,'param2',value2,...): 设置遗传算法的各种参数。
- options = gaoptimset(olddopts,'param1',value1,...): 修正旧的参数设置 olddopts 中的相应参数。
- options = gaoptimset(olddopts,newopts): 使用新的参数设置替代旧的参数设置。

表 19.1 遗传算法默认参数的设置情况

参数名	描述	默认值
PopulationType	种群类型，即适应度函数的输入数据类型	'doubleVector'
PopInitRange	初始种群的向量范围	[2x1 double]
PopulationSize	种群大小，即种群中的个体数	20
EliteCount	下一代的种群存活个数	2
CrossoverFraction	产生子代的交叉函数	0.800000000000000
MigrationDirection	变异方向	'forward'
MigrationInterval	变异间隔	20
MigrationFraction	变异概率	0.200000000000000
Generations	算法终止的最大代数	100
TimeLimit	算法终止的时间限制	Inf
FitnessLimit	算法终止的适应度的要求	-Inf
StallGenLimit	算法停滞的代数限制	50
StallTimeLimit	算法停滞的时间限制	20
InitialPopulation	指定初始种群	[]
InitialScores	指定初始种群的适应度值	[]
PlotInterval	绘图的间隔代数	1
CreationFcn	创建初始种群的函数	@gacreationuniform
FitnessScalingFcn	适应度比例函数	@fitscalingrank
SelectionFcn	选择函数	@selectionstochunif
CrossoverFcn	交叉函数	@crossoverscattered
MutationFcn	变异函数	@mutationgaussian
HybridFcn	混合函数	[]
Display	遗传算法结果显示方式	'final'
PlotFcns	绘图函数	[]
OutputFcns	输出函数	[]
Vectorized	矢量化	'off'

表 19.2 遗传算法可以设置的参数及其可以设置的范围

参数名	取值范围
PopulationType (种群的数据类型)	['bitstring' 'custom' {'doubleVector'}]
PopInitRange (种群范围)	[matrix {[0;1]}]
PopulationSize (种群大小)	[positive scalar {20}]
EliteCount (下一代的种群存活个数)	[positive scalar {2}]
CrossoverFraction (交叉概率)	[positive scalar {0.8}]
MigrationDirection (变异方向)	['both' {'forward'}]
MigrationInterval (变异区间)	[positive scalar {20}]
MigrationFraction (变异概率)	[positive scalar {0.2}]
Generations (算法的最大执行代数)	[positive scalar {100}]

(续表)

参数名	取值范围
TimeLimit (算法的时间限制)	[positive scalar {Inf}]
FitnessLimit (算法的适应度限制)	[scalar {-Inf}]
StallGenLimit (算法的停滞代数)	[positive scalar {50}]
StallTimeLimit (算法的停滞时间)	[positive scalar {20}]
InitialPopulation (指定的初始种群)	[matrix {}]
InitialScores (指定的初始种群的适应度值)	[column vector {}]
CreationFcn (创建初始种群函数)	[function_handle {@gacreationuniform}]
FitnessScalingFcn (适应度比例函数)	[function_handle @fitscalingshiftlinear @fitscalingprop @fitscalingtop @fitscalingrank]
SelectionFcn (选择函数)	[function_handle @selectionremainder @selectionrandom @selectionroulette @selectiontournament {@selectionstochunif}]
CrossoverFcn (交叉函数)	[function_handle @crossoverheuristic @crossoverintermediate @crossoversinglepoint @crossoverwopoint {@crossoversscattered}]
MutationFcn (变异函数)	[function_handle @mutationuniform {@mutationgaussian}]
HybridFcn (混合函数)	[@fminsearch @patternsearch @fminunc {}]
Display (算法的显示)	[off iter diagnose {final}]
OutputFcns (输出函数)	[function_handle @gaoutputgen {}]
PlotFcns (绘图函数)	[function_handle @gaplotbestf @gaplotbestindiv @gaplotdistance @gaplotexpectation @gaplotgeneology @gaplotselection @gaplotrange @gaplotscorediversity @gaplotscores @gaplotstopping {}]
PlotInterval (绘图的间隔)	[positive scalar {1}]
Vectorized (算法的矢量化)	['on' 'off']

3. gaoptimget()函数

函数 gaoptimget()用于获取遗传算法参数 options 结构中的各参数的具体值,其调用格式如下。

val = gaoptimget(options, 'name'): 参数 options 为算法参数的结构体变量, name 为需要获取的参数名称,返回参数值 val。

【例 19.1】利用遗传算法求解非线性规划问题。

$$\max f(x) = x \sin(10 * x) + 2, -5 < x < 5$$

(1) 在当前路径下创建遗传算法的适应度函数。

```
function f=gal(x)
if (x<=-5 | x>=5)
    f=10000;
else
    f=-x.*sin(10*x)+2;
end
```

(2) 遗传算法参数设置。

```
>> options=gaoptimset('Generations',1000,'PlotFcns',@gaplotscores) %算法参数设置
options =
    PopulationType: 'doubleVector'
      PopInitRange: [2x1 double]
    PopulationSize: 20
      EliteCount: 2
CrossoverFraction: 0.800000000000000
MigrationDirection: 'forward'
  MigrationInterval: 20
MigrationFraction: 0.200000000000000
      Generations: 1000
      TimeLimit: Inf
    FitnessLimit: -Inf
```



```

-0.00025552692465
1.000000000000000
-0.00025552692465
0.00035397349718
1.000000000000000
0.00041136706323
-0.00025552692465
1.000000000000000
-0.00025552692465
-0.00018389807315
0.00067021052006
-0.00005751042131
1.000000000000000

```

执行上述程序将生成如图 19.1 所示的遗传算法适应度的变化图。

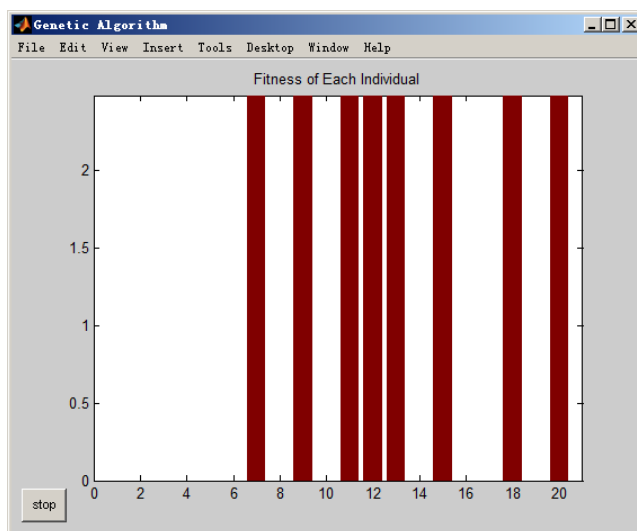


图 19.1 遗传算法适应度的变化

【例 19.2】利用遗传算法求解函数的最大值。

计算函数 $f(x,y)=0.5-(\sqrt{x^2+y^2}-0.3)/(1+0.05(x^2+y^2)^3)$ 的最大值。

(1) 在当前路径下创建遗传算法的适应度函数。

```

function f=ga2(x)
f=0.5-(sqrt(x(1)^2+x(2)^2)-0.3)/(1+0.05*(x(1)^2+x(2)^2)^3);
end

```

(2) 利用遗传算法寻找函数的最大值。

```

>> [x,fval,reason,output,population,scores] = ga(@ga2,2)
最大值
Optimization terminated: stall generations limit exceeded.
x =
    0.38160477896527   -1.27256366499624
fval =
   -0.30674405004164
reason =
Optimization terminated: stall generations limit exceeded.
output =
    randstate: [35x1 double]
    randnstate: [2x1 double]
    generations: 84
    funccount: 1680
    message: [1x58 char]
population =

```

%利用遗传算法寻找目标函数

```
0.38160477896527 -1.27256366499624
0.38160477896527 -1.27256366499624
0.38160477896527 -1.31011419822679
0.38160477896527 -1.27256366499624
0.38160477896527 -1.18973815384375
0.38160477896527 -1.27256366499624
0.38160477896527 -1.27256366499624
0.38160477896527 -1.27256366499624
0.50449157672674 -1.27256366499624
0.38160477896527 -1.27256366499624
0.38160477896527 -1.27256366499624
0.38160477896527 -1.27256366499624
0.73483065779623 -1.27256366499624
0.38160477896527 -1.27256366499624
0.38160477896527 -1.27256366499624
0.38160477896527 -1.27256366499624
0.11785579778149 -1.48001823960520
0.72983293299978 -1.20350572900787
0.31060234995628 -1.26776432607806
0.33109743981799 -1.11610835223272
scores =
-0.30674405004164
-0.30674405004164
-0.30478132889451
-0.30674405004164
-0.29769931534843
-0.30674405004164
-0.30674405004164
-0.30674405004164
-0.30674405004164
-0.30428501883637
-0.30674405004164
-0.30674405004164
-0.30674405004164
-0.30674405004164
-0.27786369725451
-0.30674405004164
-0.30674405004164
-0.30674405004164
-0.27151225692194
-0.29748274563777
-0.30597522745091
-0.26851854935123
```

19.2.2 利用 GUI 实现遗传算法

通过前面的介绍读者可以基本掌握如何通过编写命令行语句的方式实现遗传算法，但是对于一些不善于编程的用户来说，利用 GUI 实现遗传算法显得更为高效。本小节主要介绍如何利用 GUI 实现遗传算法。

1. 遗传算法工具箱 GUI 的启动

遗传算法工具箱 GUI 的启动主要有下面两种方法。

- 启动 MATLAB 软件，在命令窗口内输入 `gatool` 命令。
- 选择 MATLAB 主菜单中的“Start”→“Toolboxes”→“Genetic Algorithm and Direct Search Toolbox ”→“Genetic Algorithm Tool”命令。

使用上述任意操作将打开如图 19.2 所示的遗传算法工具箱 GUI。

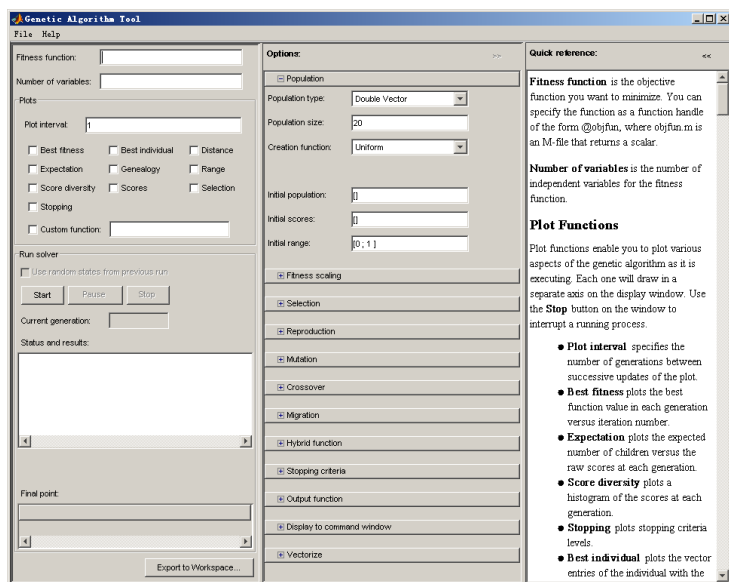


图 19.2 遗传算法工具箱 GUI

2. 遗传算法工具箱 GUI 的工作界面

遗传算法工具箱的 GUI 仅包括“File”和“Help”两个菜单项。“Help”菜单项主要用于获取遗传算法工具箱的使用帮助，而“File”用于算法中数据的导入/导出等，其主要菜单项如图 19.3 所示，其中，

- “Import Options”菜单项：导入遗传算法的参数数据。
- “Import Problem”菜单项：导入遗传算法需要求解的问题变量，Problem 为结构体变量，一般包含适应度函数，变量个数和算法参数。
- “Export to Workspace”菜单项：导出遗传算法计算的结果到 MATLAB 工作空间中。
- “Generate M-File”菜单项：遗传算法的 GUI 操作转换为 M 文件，生成相应的代码。
- “Close”菜单项：关闭工具箱。

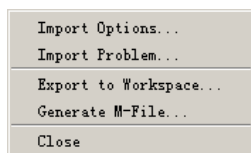


图 19.3 遗传算法工具箱 GUI 的“File”菜单项

遗传算法工具箱的 GUI 的主界面中包含了 3 个区域，从左到右依次为遗传算法的实现、算法参数设置和算法简单的帮助文档。

1) 遗传算法的实现区域

遗传算法的实现区域主要包括以下部分。

- Fitness function：输入适应度函数。
- Number of variables：输入适应度函数的输入变量个数。
- Plots：控制遗传算法求解中的绘图部分，包括绘图的间隔和绘图的类型。
- Run solver：控制遗传算法的执行，包括 Start（启动）、Pause（暂停）和 Stop（终止）。
- Status and results：显示遗传算法的执行状态和算法的结果。
- Final point：遗传算法最终的结果。

2) 遗传算法参数设置区域

这部分区域主要用于设置遗传算法的参数: Population (群体)、Fitness scaling (拟合的比例系数)、Selection (选择)、Reproduction (繁殖)、Mutation (变异)、Crossover (交叉)、Migration (迁移)、Hybrid function (混合函数)、Stopping criteria (终止的条件)、Output function (输出函数)、Display to command window (命令窗口的显示) 和 Vectorize (矢量化)。单击这些参数前面的加号 “+”, 将可具体展开各参数的设置区域, 用于在其中算法相应的参数。

3) 算法的帮助文档

MATLAB 遗传算法工具箱 GUI 的最右面为遗传算法的简单的帮助文档, 供用户在使用遗传算法工具箱的时候快速浏览, 获取帮助信息。如果用户不希望帮助文档在使用遗传算法工具箱 GUI 的时候显示, 可以单击 “Quick reference” 右边的 “<<” 符号, 不显示帮助文档。同时, 需要显示的时候单击 “Options” 区域右侧的 “>>” 符号即可。

3. 利用 GUI 实现遗传算法的实例分析

通过前面的简单介绍, 读者应该对遗传算法工具箱的 GUI 有了初步的了解, 其大致的功能同通过函数法实现遗传算法, 主要的不同之处在于利用 GUI 时无须编写复杂的代码, 各参数的设置也很方便。下面以本章例 19.1 的问题为例, 使用 GUI 工具箱来实现这个问题。

【例 19.3】利用 GUI 实现遗传算法。

(1) 在 MATLAB 命令窗口中输入 `gatool` 命令, 打开遗传算法工具箱的 GUI。

(2) 在遗传算法工具箱 GUI 中设置遗传算法。

- 在 “Fitness function” 文本框中输入 “@ga”。
- 在 “Number of variables” 文本框中输入 “1”。
- 在 “Plot” 区域中选择 “Best fitness” 复选框。

(3) 单击 “Run solver” 区域中的 “Start” 按钮, 启动遗传算法。

(4) 算法运行完毕, 将生成如图 19.4 所示的遗传算法最佳适应度的图, 在 “Status and results” 区域显示运行的结果, 并在 “Final point” 显示最终的遗传算法的结果, 如图 19.5 所示。

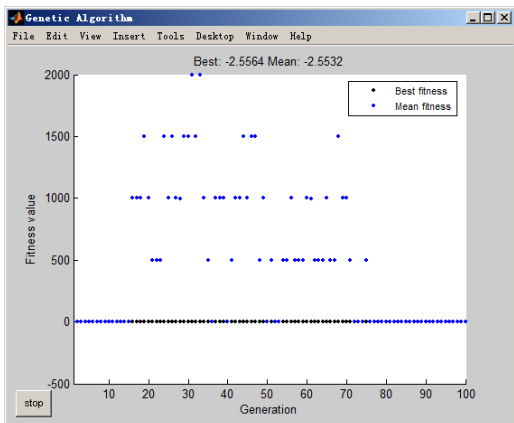


图 19.4 遗传算法最佳适应度

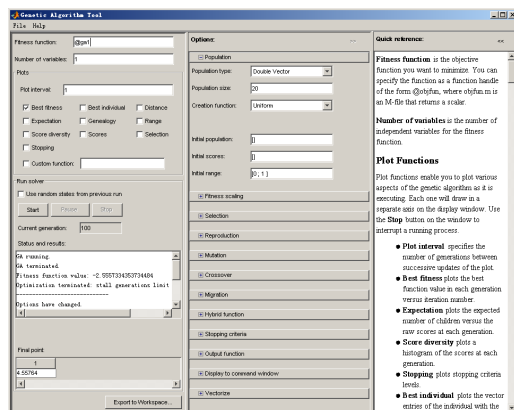


图 19.5 遗传算法的结果显示

19.3 直接搜索工具箱

直接搜索工具箱可以按照一定的模式搜寻函数的最小值。本节主要介绍 MATLAB 直接搜索工

工具箱的使用。直接搜索工具箱的使用方法也可以分为函数法和 GUI 法，下面具体展开叙述。

19.3.1 利用命令行方式实现直接搜索

MATLAB 直接搜索工具箱的几个主要函数为 `patternsearch()`、`psoptimget()` 和 `psoptimset()`，下面具体介绍这几个函数的使用。

1. `patternsearch()` 函数

函数 `patternsearch()` 可用于模式搜索，其调用格式如下。

- `x = patternsearch(@fun, x0)`: 无约束条件的模式搜索，其中输入参数 `@fun` 为待搜索的函数的句柄，`x0` 为模式搜索的起点，返回函数的最小值点 `x`。
- `x = patternsearch(@fun, x0, A, b)`: 有约束条件的模式搜索，约束条件为 $Ax \leq b$ 。
- `x = patternsearch(@fun, x0, A, b, Aeq, beq)`: 有约束条件的模式搜索，约束条件为 $Ax \leq b$ 和 $Aeqx = beq$ 。
- `x = patternsearch(@fun, x0, A, b, Aeq, beq, lb, ub)`: 有约束条件的模式搜索，约束条件为 $Ax \leq b$ 、 $Aeqx = beq$ 和 $lb \leq x \leq ub$ 。
- `x = patternsearch(@fun, x0, A, b, Aeq, beq, lb, ub, options)`: 参数 `options` 用于设置模式搜索算法的参数，算法的参数可以通过函数 `psoptimset()` 设置。
- `x = patternsearch(problem)`: 以 `problem` 结构体变量的形式存放模式搜索函数的输入参数。
- `[x, fval] = patternsearch(@fun, x0, ...)`: 返回函数的最小值点 `x`，和最小点处的函数值。
- `[x, fval, exitflag] = patternsearch(@fun, x0, ...)`: 参数 `exitflag` 为模式搜索算法终止的状况。
- `[x, fval, exitflag, output] = patternsearch(@fun, x0, ...)`: 输出参数 `output` 为模式搜索算法的相关信息。

2. `psoptimget()` 函数

函数 `psoptimget()` 可用于获取模式搜索算法的参数，其调用格式如下。

`val = psoptimget(options, 'name')`: 返回参数 `name` 的值。

3. `psoptimset()` 函数

函数 `psoptimset()` 用于设置算法的相应参数，其调用格式基本格式同遗传算法参数设置的函数 `gaoptimset()`，在此不再详细展开叙述，可以设置的具体参数参考 MATLAB 的帮助文档。

【例 19.4】 利用函数命令行实现模式搜索。

有约束条件：

```
5x1+3x2+2x3≤3
3x1+5x2+2x3≤5
2x1+3x2+x3≤8
x1≥0, x2≥0, x3≥0
```

计算函数 $\sin(x(1)^2+x(2)^2-x(3)^2)+5$ 的最小值。

(1) 在当前目录下创建待搜索的函数。

```
function y=pat(x)
y=sin(x(1)^2+x(2)^2-x(3)^2)+5;
end
```

(2) 执行有约束条件的模式搜索。

```
>> A = [5 3 2; 3 5 2; 2 3 1];
b = [8; 5; 2];
lb = zeros(3,1);
[x, fval, exitflag, output] = patternsearch(@pat,[0 0 0],A,b,[],[],lb) % 约束条件的
模式搜索
Optimization terminated: current mesh size 9.5367e-007 is less than 'TolMesh'.
```

```
x =  
    0.02253913879395    0.03125000000000    1.25390625000000  
fval =  
    4  
exitflag =  
    1  
output =  
    function: @pat  
    problemtype: 'linearconstraints'  
    pollmethod: 'positivebasis2n'  
    searchmethod: []  
    iterations: 44  
    funccount: 189  
    meshsize: 9.536743164062500e-007  
    message: [1x78 char]
```

19.3.2 利用 GUI 方式实现模式搜索

前面的小节中介绍了如何利用函数法实现模式搜索，本小节主要介绍如何利用 GUI 实现模式搜索算法。

1. 模式搜索工具箱的启动

模式搜索工具箱 GUI 的启动主要有下面两种方法。

- 启动 MATLAB 软件，在命令窗口内输入 psearchtool 命令。
- 选择 MATLAB 主菜单中的 “Start” → “Toolboxes” → “Genetic Algorithm and Direct Search Toolbox” → “Direct Search Tool” 命令。

使用上述任意操作将打开如图 19.6 所示的模式搜索工具箱 GUI。

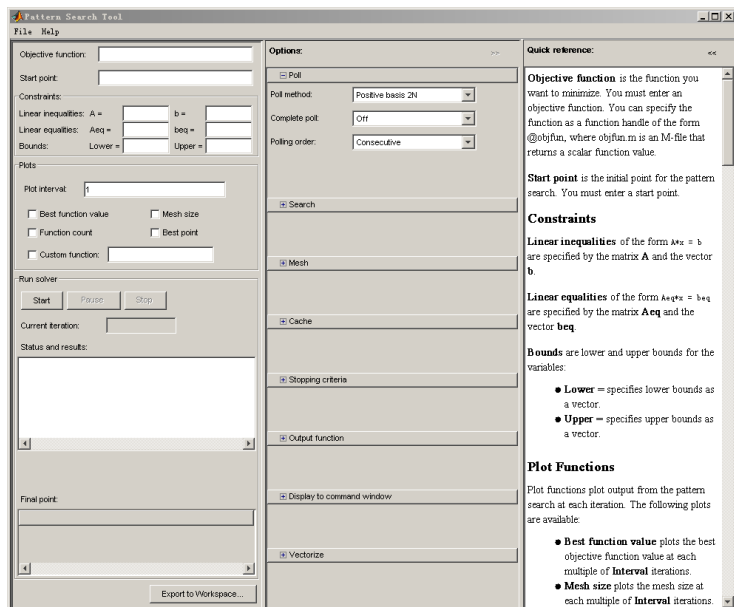


图 19.6 模式搜索工具箱 GUI

2. 模式搜索工具箱 GUI 的工作界面

模式搜索工具箱 GUI 的工作界面与遗传算法工具箱的 GUI 类似，也包括 “File” 和 “Help” 两个菜单项，且其界面环境也与遗传算法的界面环境类似，包含了算法控制、算法参数设置和算法帮助部分。模式搜索工具箱 GUI 的工作界面各区域的功能也与遗传算法的类似，这里不再详细

展开叙述,读者可以参考遗传算法工具箱 GUI 的介绍和帮助文档进行学习。下面演示如何利用模式搜索工具箱 GUI 实现例 19.5。

【例 19.5】利用 GUI 实现模式搜索。

求解例 19.5 中的问题。

(1) 在 MATLAB 命令窗口中输入 psearchtool 命令,打开模式搜索工具箱 GUI。

(2) 在模式搜索工具箱 GUI 中设置模式搜索。

- 在“Objective function”文本框中输入“@pat”。
- 在“Start point”文本框中输入“[0 0 0]”。
- 在“Constraints”中输入“A=[5 3 2;3 5 2;2 3 1]”和“b=[8;5;2]”。
- 在“Plot”区域中选择“Best Function Value”复选框。

(3) 单击“Run solver”区域中的“Start”按钮,启动模式搜索算法。

(4) 算法运行完毕,将生成如图 19.7 所示的模式搜索算法函数最佳值的变化图,在“Status and results”区域显示运行的结果,并在“Final point”显示最终的遗传算法的结果,如图 19.8 所示。

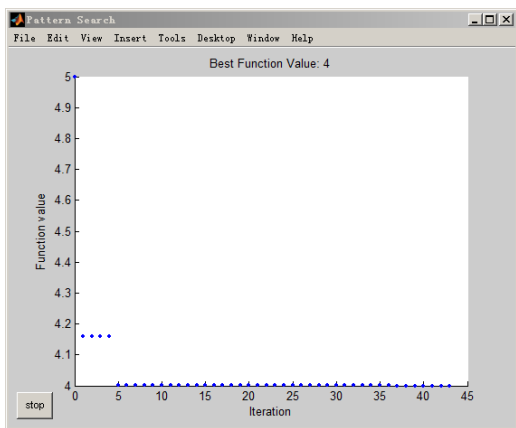


图 19.7 模式搜索算法函数最佳值的变化

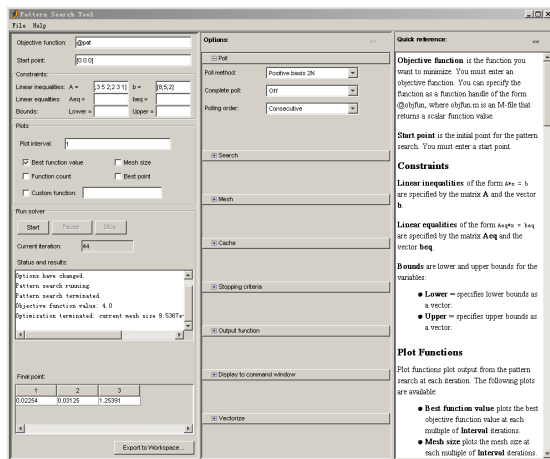


图 19.8 模式搜索算法结果的显示

19.4 本章小结

本章主要介绍了 MATLAB 遗传算法和直接搜索工具箱的使用。本章的内容主要分为遗传算法和直接搜索两部分,其中重点讲述了遗传算法工具箱的使用,而直接搜索工具箱与遗传算法工具箱的使用有很大的相同之处,读者可以触类旁通。遗传算法和模式搜索算法的实现都可以分为函数命令行法和图形界面法, MATLAB 中许多工具箱都为读者提供了这两种不同的操作模式,在实际的操作中用户可以根据自身的实际掌握能力和问题的复杂性采用相应的求解方法。

第20章

MATLAB 在各领域的应用

通过前面章节的介绍，相信读者已经发现 MATLAB 软件强大的数据处理和图像分析功能，同时其丰富的工具箱也为用户在求解复杂算法的时候提供了较为方便的工具。但是相信更多的用户在使用 MATLAB 的时候主要致力于解决的还是其研究领域的问题，因而本章将简单介绍 MATLAB 软件在数学建模、物理、化学等领域的应用，希望能给使用 MATLAB 软件的用户一些启示。

20.1 MATLAB 在数学建模中的应用

在生产实践中许多复杂的问题都可以通过数学模型来量化，而数学建模的过程一般包括：（1）通过调查研究获取数据资料；（2）分析数据资料，抽象出研究对象；（3）提出合理的假设，分析内在的定量关系；（4）使用简单的数学符号定义问题；（5）求解模型。其中 MATLAB 软件在数学模型的求解中可以发挥很大的作用，特别是对于一些复杂的模型，人工求解费时费力。特别是对于数学建模竞赛，中国大学生数学建模竞赛乃至国际各大数学建模竞赛一般都有时间限制，要求参赛者在 72 小时内完成建模工作，因而如何能方便、高效的求解出模型在数学建模中相当重要。而 MATLAB 软件无疑为广大用户提供了一个方便、友好、高效的求解工具。

下面我们以中国人口增长预测模型的建立为例演示 MATLAB 在数学建模中的应用，如表 20.1 所示为 1995—2005 年中国人口数的变化。

【例 20.1】建立中国人口增长的数学模型，并由此对中国人口增长的中短期和长期趋势做出预测。

表 20.1 1995—2005 年中国人口数的变化

年 份	人口总数
1995	121121
1996	122389
1997	123626
1998	124761
1999	125786
2000	126743
2001	127627
2002	128453
2003	129227
2004	129988
2005	130756

利用 MATLAB 构建中国人口增长的一元线性回归模型： $y=at+b$ 。

```

load data;
X=[ones(size(t)) t];
[b,bint,r,rint,stats] = regress(y,X)    %一元线性回归, 返回回归系数 b, 回归系数指向区间 bint
等参数
rcoplot(r,rint)
b =
    1.0e+006 *
   -1.775392999999998
    0.000950900000000
bint =
    1.0e+006 *
   -1.91398843471870   -1.63679756528125
    0.00088160236926    0.00102019763074
r =
    1.0e+002 *
   -5.315000000000087
   -2.144000000000082
    0.716999999999924
    2.557999999999945
    3.298999999999951
    3.359999999999971
    2.690999999999977
    1.441999999999983
   -0.326999999999997
   -2.225999999999991
   -4.054999999999971
rint =
    1.0e+003 *
   -1.00529399019741   -0.05770600980277
   -0.86586017608920    0.43706017608903
   -0.62710814715546    0.77050814715531
   -0.43469081076177    0.94629081076166
   -0.35217255297547    1.01197255297538
   -0.34812503881389    1.02012503881383
   -0.42984149580445    0.96804149580440
   -0.56666735809805    0.85506735809802
   -0.73336918326186    0.66796918326186
   -0.87229852222289    0.42709852222291
   -0.95322382760779    0.14222382760785
stats =
    1.0e+005 *
Columns 1 through 3
    0.00000990746066    0.00963559350259    0.000000000000000
Column 4
    1.03224766666666

```

执行上述程序将生成如图 20.1 所示的模型残差分析图。所建立的线性回归模型为 $y = -1.775 \times 10^6 + 950.9 \times t$, 模型系数的置信区间为 $[-1.914 \times 10^6, -1.637 \times 10^6]$ 和 $[881.6, 1020.2]$, 所建立的模型的决定系数为 0.9907。利用所构建的模型预测 2006 年的人口数: $y = -1.775 \times 10^6 + 950.9 \times 2006 = 132505.4$ 。

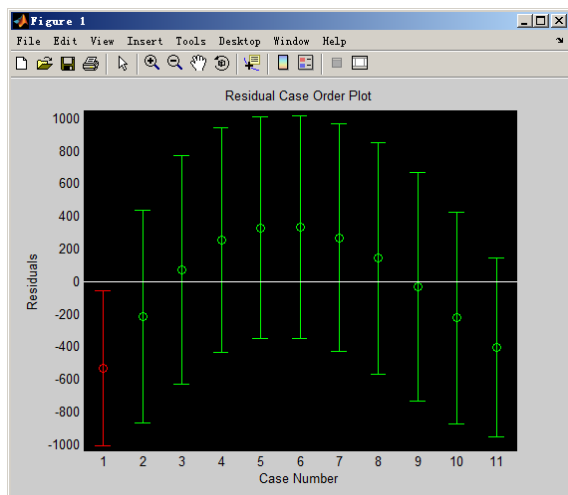


图 20.1 模型残差分析图

综上，我们可以看到在数学建模中应用 MATLAB 软件可以避免复杂的编程，可以较大地提高数学模型的求解速度，让用户可以把更多的精力放在模型的设计和优化上，同时 MATLAB 还提供了丰富的方法可用于数学模型的构建。

20.2 MATLAB 在物理中的应用

物理问题一般都比较复杂，需要大量复杂公式的证明，方程求解。应用 MATLAB 可以把一些复杂的物理问题通过编程方便地利用计算机求解，同时解的可视化也较为方便，有助于提高我们分析处理实际的物理问题的能力。

数学物理学是不少理工科学生都要学习的课程，其中涉及许多复杂物理问题的数学求解，十分烦琐，也不便于理解。但是用户借助 MATLAB 强大的数据分析能力和图像显示能力将可以方便地理解、处理傅里叶变换、热传导方程、泊松方程等物理问题。

下面将以几个具体的实例演示 MATLAB 在物理中的应用。

【例 20.2】求解泊松方程并与精确解比较。

```
g='circleg';
b='circleb1';
c=1;
a=0;
f=1;
[p,e,t]=initmesh(g,'hmax',1); %根据输入参数 g 所代表的求解区域几何形状，返回三角网格数据
error=[]; er=1;
while er > 0.001 %求解过程中误差的循环，当误差小于等于 0.001 后停止循环
    [p,e,t]=refinemesh(g,p,e,t);
    u=assemblpde(b,p,e,t,c,a,f);
    exact=(1-p(1,:).^2-p(2,:).^2)/4;
    er=norm(u-exact,'inf');
    error=[error er];
    fprintf('Error: %e. Number of nodes: %d\n',er,size(p,2));...
    [p,e,t]=refinemesh(g,p,e,t);
    u=assemblpde(b,p,e,t,c,a,f);
    exact=(1-p(1,:).^2-p(2,:).^2)/4;
    er=norm(u-exact,'inf');
    error=[error er];
    fprintf('Error: %e. Number of nodes: %d\n',er,size(p,2));
    [p,e,t]=refinemesh(g,p,e,t);
```



```

u=asmpde(b,p,e,t,c,a,f);
exact=(1-p(1,:).^2-p(2,:).^2)'/4;
er=norm(u-exact,'inf');
error=[error er];
fprintf('Error: %e. Number of nodes: %d\n',er,size(p,2));
[p,e,t]=refinemesh(g,p,e,t);
u=asmpde(b,p,e,t,c,a,f);
exact=(1-p(1,:).^2-p(2,:).^2)'/4;
er=norm(u-exact,'inf');
error=[error er];
fprintf('Error: %e. Number of nodes: %d\n',er,size(p,2));
end
pdesurf(p,t,u); %绘制泊松方程的动态解
figure;
pdesurf(p,t,u-exact); %绘制泊松方程的精确解与仿真解的误差解

```

执行上述程序将生成如图 20.2 所示的泊松方程的动态解和图 20.3 所示的泊松方程的精确解与仿真解的误差解。

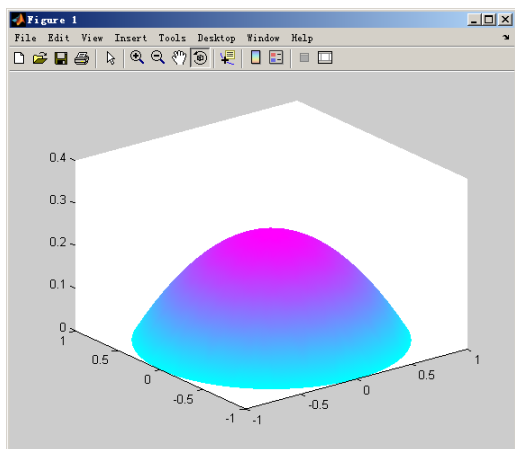


图 20.2 泊松方程的动态解

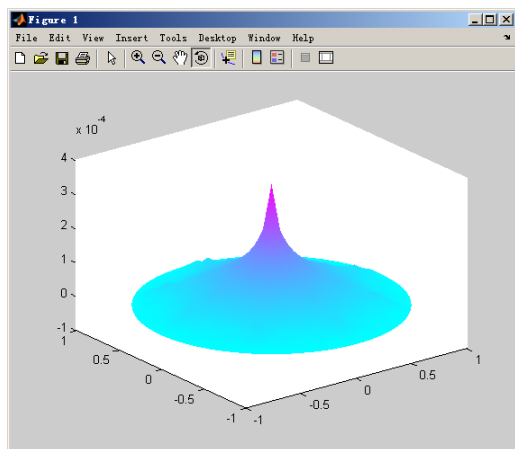


图 20.3 泊松方程的精确解与仿真解的误差解

【例 20.3】绘制第一类贝塞尔曲线 $J_{10}(x)$ 。

```

x=0:0.2:100
y=besselj(10,x)
plot(x,y)

```

执行上述程序将生成如图 20.4 所示的第一类贝塞尔曲线。

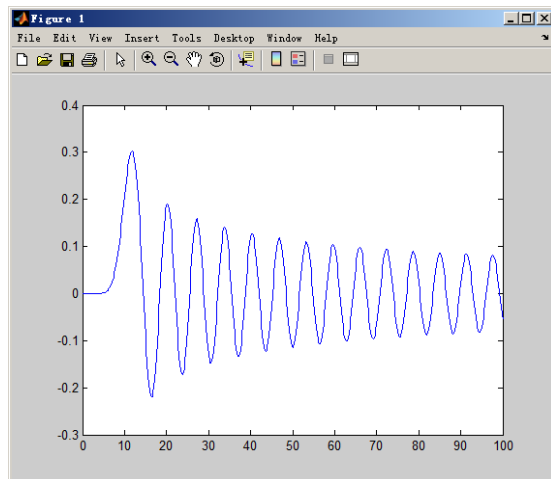


图 20.4 第一类贝塞尔曲线

20.3 MATLAB 在化学中的应用

在前面简单介绍了 MATLAB 在数学、物理中的应用,其在化学领域也有较好的表现。MATLAB 在化学中的应用主要体现在以下几个方面。

- 化学、化工过程的动态模拟。
- 化学实验数据的分析处理。
- 化学计量学的应用。

下面以一些具体的实例演示 MATLAB 在化学中的应用。

【例 20.4】 化学反应速率方程拟合。

化学反应速率的解为:

$$v = -\frac{dc}{dt} = kc^n$$

现有一化学反应,在不同时间下溶液中物质的浓度如表 20.2 所示,试拟合该化学反应的速率方程。

表 20.2 化学反应不同时间下溶液中物质的浓度

t/s	0	5	10	15	20	25	30
c/(mol/l)	13.5	11.25	9.52	7.93	6.98	6.06	5.56

根据表 20.2 中的数据,拟合该化学反应的速率方程。

```
>> t=0:5:30;
c=[13.5 11.25 9.52 7.93 6.98 6.06 5.56]
d=(max(t)-min(t))/10000;
x=min(t):d:max(t);
y=interp1(t,c,x,'cubic');           %数据插值
v=diff(y)./diff(x);
log(-v);
m=length(y)-1;
logc=log(y(1:m));
nk=polyfit(logc,log(-v),1);           %曲线拟合
n=nk(1);
k=exp(nk(2));
n
k
c =
Columns 1 through 3
13.50000000000000 11.25000000000000 9.52000000000000
Columns 4 through 6
7.93000000000000 6.98000000000000 6.06000000000000
Column 7
5.56000000000000
n =
1.85235431123672
k =
0.00473636036040
```

最后,得出的化学反应速率方程为 $v=0.00474 \times c^{1.852}$ 。

【例 20.5】 化学试验数据的假设检验。

饮用水中 5 次硝酸根离子的测定值分别为 (mg/L) 52.3、51.5、50.8、49.9、48.8,在显著水平 0.05 下,测量值的均值与真值 50.0 mg/L 是否存在显著差异?

```
>> x=[52.3 51.5 50.8 49.9 48.8];
[h,p,ci,stats] = ttest(x,50)           %T 检验
```

```

h =
    0
p =
    0.34042237066803
ci =
    48.96523098048641    52.35476901951360
stats =
    tstat: 1.08124101180258
         df: 4
         sd: 1.36491757992928

```

结果 $h=0$ 即在显著水平 0.05 下, 测量值的均值与真值 50.0 mg/L 是不存在显著差异的。

20.4 MATLAB 在生命科学中的应用

近年来, 生命科学领域的试验研究为了得到更为可靠的研究结论往往需要对大量的数据进行分析处理, 建立数学模型, 寻找数据的规律。MATLAB 强大的图形处理能力在医学图像的处理上也很优势。下面以几个实例演示 MATLAB 在生命科学中的应用。

【例 20.6】在理想环境中细菌的繁殖模型为:

$$\frac{dx(t)}{dt} = 0.5x(t)$$

试求解该模型。

在当前目录下编写需要求解的微分方程的表达式。

```

function f=funtest(t,x)
f=0.5*x;
end

```

求解该模型。

```

[t,Y] = ode45(@funtest,[0 50],50);
plot(t,Y)

```

执行上述程序, 将生成如图 20.5 所示的细菌的繁殖模型的解。

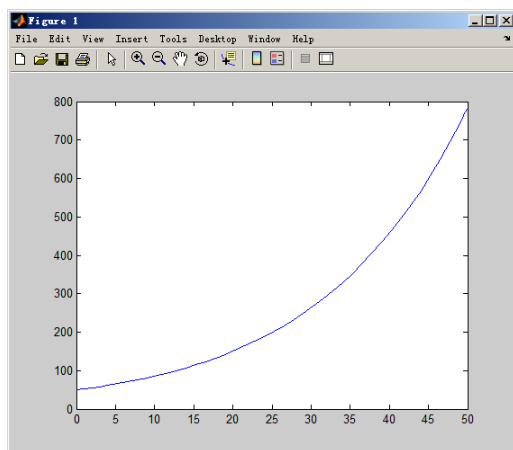


图 20.5 细菌的繁殖模型的解

【例 20.7】模型拟合。

服用一定药物后, 在体内的药物浓度与时间的关系为: $c(t)=ae^{bt}+d$, 试根据试验获得的数据求解模型的系数。

在当前目录下创建待拟合的模型。

```
function c=funtest2(beta,t)
c=beta(1)*exp(beta(2)*t)+beta(3);
end
```

利用函数 `nlinfit()` 估计非线性模型的参数。

```
>> t=0:1:15;
c=[5 4.5 4.1 3.9 3.7 3.3 3.15 3.05 2.86 2.76 2.65 2.54 2.46 2.43 2.39 2.32];
plot(t,c)
beta=[0 0 0]
betahat = nlinfit(t',c',@funtest2,beta) %非线性模型参数估计
beta =
    0    0    0
betahat =
-2.90993375050504 -0.15913047679154 2.05503168413094
```

20.5 MATLAB 在社会科学中的应用

近年来 MATLAB 除了在理工科领域有不俗的表现,在社会科学,特别是对社会调查中的数据的数据的分析处理也很有效。下面以反映美国城市生活质量的数据 `cities.mat` 为基础,应用主成分分析的方法,对该数据文件进行分析,判断对城市生活质量的主要影响因素。

【例 20.8】主成分分析在社会科学数据分析中的应用。

```
>> load cities %导入数据
who %查看数据变量
Your variables are: %导入的数据包含 3 个变量
categories names ratings % categories 为数据的类别, names 为城市名, ratings 为调查的数据
>> categories %显示数据的类别, 调查的影响生活质量的
categories =
climate
housing
health
crime
transportation
education
arts
recreation
economics
[coeff, score, latent, tsquare] = princomp(ratings); %主成分分析
percent_explained = 100*latent/sum(latent); %贡献率计算
pareto(percent_explained);
xlabel('Principal Component');
ylabel('Variance Explained (%)');
biplot(coeff(:,1:2), 'scores',score(:,1:2),'varlabels',categories);
```

执行上述程序,将生成如图 20.6 所示的主成分分析的数据盒形图、图 20.7 所示的主成分分析的各主成分的共享率分布图和图 20.8 所示的主成分分析第一和第二主成分的得分分布图。

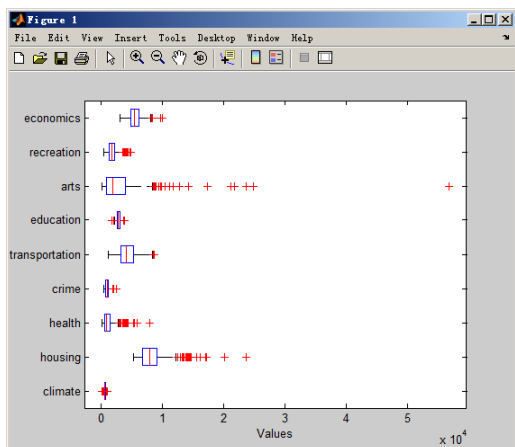


图 20.6 主成分分析的数据盒形图

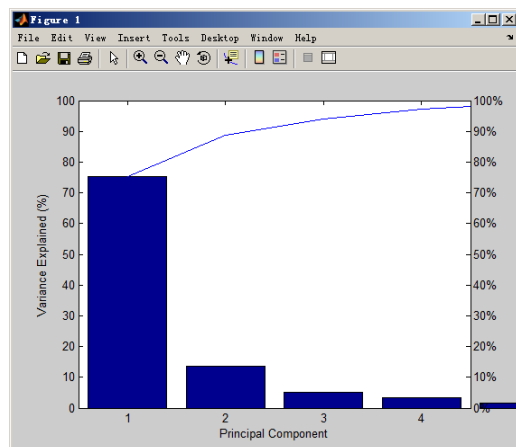


图 20.7 主成分分析的各主成分的共享率分布图

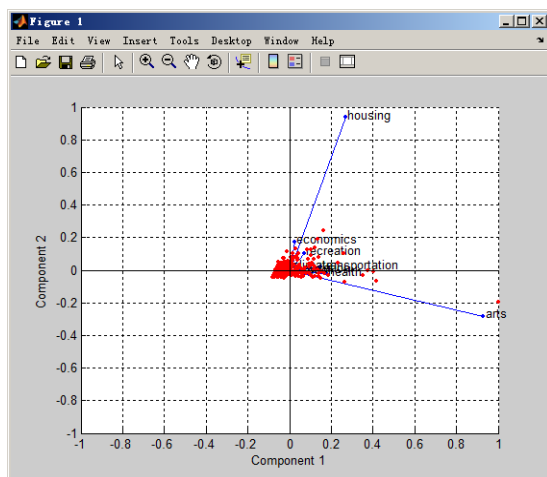


图 20.8 主成分分析第一和第二主成分的得分分布图

20.6 本章小结

本章主要介绍了 MATLAB 软件在数学建模、物理、化学、生命科学、社会科学领域的应用，同时 MATLAB 在其他许多领域也有着广泛的应用。希望通过本章每个领域简单的实例演示能给广大读者一些启示，在其自身的研究领域中发挥 MATLAB 强大的作用。

本书特色

- 内容全面、翔实，涉及MATLAB各项功能和常用工具箱
- 由浅入深，适合于各个层次的科学工作者
- 每个函数结合实例讲解，容易掌握
- 介绍了实际操作中可能遇到的问题
- 面向需求，对常用工具箱进行了详细介绍
- 能满足各研究领域实际问题的解决
- 结合编程和图形用户界面（GUI）两种模式讲解

MATLAB 从基础到精通

本书涵盖内容

MATLAB概述、安装和学习方法

MATLAB的数据类型

矩阵和数组

程序设计

图形处理

图形用户界面 (GUI)

数值分析

符号计算功能

应用程序接口

文件I/O

Simulink仿真

统计工具箱

图像处理工具箱

优化工具箱

曲线拟合工具箱

神经网络工具箱

金融工具箱

小波分析工具箱

遗传算法工具箱

MATLAB在各领域的应用

本书读者对象

MATLAB入门读者

以MATLAB为工具的研究人员

大中专院校相关专业的学生和老师

本书配套光盘内容

本书多媒体教学视频

本书教学PPT

本书涉及的实例文件

上架建议: 计算机辅助设计 > MATLAB

ISBN 978-7-121-15651-9



9 787121 156519 >

定价: 65.00元(含DVD光盘1张)



策划编辑: 胡辛征
责任编辑: 李云静
封面设计: 侯士卿